Program MARK

A Gentle Introduction

EVAN G. COOCH & GARY C. WHITE (eds.)

❖ 13th edition

Table of contents

1 First steps. . .
- 1.1 Return 'rates' . . . 1 - 2
- 1.2 A more robust approach . . . 1 - 3
- 1.3 Maximum likelihood theory - the basics . . . 1 - 5
  - 1.3.1 Why maximum likelihood? . . . 1 - 5
  - 1.3.2 Simple estimation example - the binomial coefficient . . . 1 - 6
  - 1.3.3 Multinomials: a simple extension . . . 1 - 13
- 1.4 Application to mark-recapture . . . 1 - 15
- 1.5 Variance estimation for > 1 parameter . . . 1 - 19
- 1.6 More than 'estimation' - ML and statistical testing . . . 1 - 20
- 1.7 Technical aside: a bit more on variances . . . 1 - 22
- 1.8 Summary . . . 1 - 23

2 Data formatting: the input file . . .
- 2.1 Encounter histories formats . . . 2 - 2
  - 2.1.1 Groups within groups... . . . 2 - 4
- 2.2 Removing individuals from the sample . . . 2 - 4
- 2.3 Different encounter history formats . . . 2 - 5
- 2.4 Some more examples . . . 2 - 7
  - 2.4.1 Dead recoveries only . . . 2 - 7
  - 2.4.2 Individual covariates . . . 2 - 8

Addendum: generating .inp files . . . 2 - 10

3 First steps in using Program MARK. . .
- 3.1 Starting **MARK** . . . 3 - 1
- 3.2 starting a new project . . . 3 - 2
- 3.3 Running the analysis . . . 3 - 7
- 3.4 Examining the results . . . 3 - 9
  - 3.4.1 **MARK**, PIMs, and parameter indexing . . . 3 - 10
- 3.5 Summary . . . 3 - 17

4 Building & comparing models
- 4.1 Building models - parameter indexing & model structures . . . 4 - 2
- 4.2 A quicker way to build models - the PIM chart . . . 4 - 12

4.2.1 PIM charts and single groups - Dipper re-visited . . . 4 - 12
4.2.2 PIM charts and multiple groups . . . 4 - 19
4.3 Model selection - the basics . . . 4 - 29
4.3.1 The AIC, in brief... . . . 4 - 30
4.3.2 Some important refinements to the AIC . . . 4 - 33
4.3.3 BIC - an alternative to the AIC . . . 4 - 35
4.4 Using the AIC for model selection - simple mechanics... . . . 4 - 37
4.4.1 AIC weights and 'rules-of-thumb' for model selection . . . 4 - 39
4.5 Model uncertainty: an introduction to model averaging . . . 4 - 40
4.5.1 Model averaging: deriving SE and CI values . . . 4 - 48
4.6 Significance? . . . 4 - 50
4.6.1 Classical significance testing in **MARK** . . . 4 - 51
4.6.2 Some problems with the classical approach... . . . 4 - 55
4.6.3 'Significance' of a factor using AIC . . . 4 - 57
4.7 LRT or AIC? . . . 4 - 59
4.8 Summary . . . 4 - 60
Addendum: counting parameters . . . 4 - 61

5 Goodness of fit testing...
5.1 Conceptual motivation - 'c-hat' ($\hat{c}$) . . . 5 - 2
5.2 The practical problem - estimating $\hat{c}$ . . . 5 - 6
5.3 Program **RELEASE** - details, details... . . . 5 - 7
5.4 Program Release - TEST 2 & TEST 3 . . . 5 - 7
5.4.1 Running **RELEASE** . . . 5 - 9
5.4.2 Running **RELEASE** as a standalone application . . . 5 - 12
5.5 Enhancements to **RELEASE** - program **U-CARE** . . . 5 - 17
5.5.1 **RELEASE** & **U-CARE** – estimating $\hat{c}$ . . . 5 - 21
5.6 **MARK** and bootstrapped GOF testing . . . 5 - 24
5.6.1 **RELEASE** versus the bootstrap . . . 5 - 28
5.7 'median $\hat{c}$' - a way forward? . . . 5 - 28
5.8 What to do when the general model 'doesn't fit'? . . . 5 - 32
5.8.1 Inappropriate model . . . 5 - 32
5.8.2 Extra-binomial variation . . . 5 - 35
5.9 How big a $\hat{c}$ is 'too big?' . . . 5 - 37
5.9.1 General recommendations . . . 5 - 38
5.10 Summary . . . 5 - 40

6 Adding constraints: MARK and linear models
6.1 A (brief) review of linear models . . . 6 - 1
6.2 Linear models and the 'design matrix': the basics . . . 6 - 7
6.3 The European Dipper – the effects of flooding . . . 6 - 12
6.3.1 Design matrix options: full, reduced, and identity . . . 6 - 21
6.4 Running the model: details of the output . . . 6 - 22
6.5 Reconstituting parameter values . . . 6 - 25
6.5.1 Subset models and the design matrix . . . 6 - 33
6.6 Some additional design matrix tricks . . . 6 - 37
6.7 Design matrix...or PIMs . . . 6 - 38

6.8 Constraining with 'real' covariates . . . 6 - 40
6.8.1 A special case of 'real covariates' - linear trend . . . 6 - 44
6.9 More than 2 levels of a group . . . 6 - 52
6.10 > 1 classification variables: *n*-way ANOVA . . . 6 - 54
6.11 Time and Group – building additive models . . . 6 - 58
6.12 Linear models and 'effect size': a test of your understanding . . . 6 - 61
6.12.1 Linear models: β estimates and odds ratios . . . 6 - 69
6.12.2 $\hat{c}$ and effect size: a cautionary note . . . 6 - 70
6.13 Pulling all the steps together: a sequential approach . . . 6 - 71
6.14 A final example: mean values . . . 6 - 80
6.15 RMark – an alternative approach to linear models . . . 6 - 85
6.16 Summary . . . 6 - 85

7 'Age' and cohort models...
7.1 'Age' models . . . 7 - 2
7.2 Constraining an age model . . . 7 - 14
7.2.1 DM with > 2 age classes: 'ugly' interaction terms . . . 7 - 22
7.3 Using data where both young and adults are marked . . . 7 - 24
7.3.1 Marked as young and adult: the design matrix . . . 7 - 26
7.4 'Time since marking' - when an age model is NOT an 'age' model . . . 7 - 32
7.4.1 Age, transience and the DM – a complex example . . . 7 - 37
7.5 Age/TSM models and GOF . . . 7 - 42
7.6 Cohort models . . . 7 - 42
7.6.1 Building cohort models: PIMS and design matrices . . . 7 - 44
7.7 Model averaging and age/cohort models . . . 7 - 47
7.8 Summary . . . 7 - 50

8 'Dead' recovery models
8.1 'Brownie' parameterization . . . 8 - 1
8.2 Counting parameters - Brownie parameterization . . . 8 - 7
8.3 Brownie estimation: individuals marked as young only . . . 8 - 10
8.4 Brownie analysis: individuals marked both as young + adults . . . 8 - 12
8.5 A different parameterization: Seber (S and r) models . . . 8 - 16
8.5.1 Seber vs. Brownie estimates in constrained models: careful! . . . 8 - 18
8.6 Recovery analysis when the number marked is not known . . . 8 - 23
8.7 Recovery models and GOF . . . 8 - 25
8.8 Summary . . . 8 - 28

9 Joint live encounter & dead recovery data
9.1 Combining live encounters and dead recoveries - first steps . . . 9 - 1
9.1.1 Estimating fidelity rate, $\mathbf{F_i}$: some key assumptions . . . 9 - 2
9.2 Survival, fidelity and encounter probability: probability structure . . . 9 - 3
9.3 Combined recapture/recovery analysis in **MARK**: marked as adult & young . . . 9 - 5
9.4 Marked as young only: combining live encounters & dead recoveries . . . 9 - 10
9.5 Joint live-recapture/live resight/tag-recovery model (Barker's Model) . . . 9 - 11
9.6 Barker Model - Movement . . . 9 - 12
9.7 Live encounters, dead recoveries & multi-state models . . . 9 - 13
9.7.1 Barker model: assumptions . . . 9 - 14

9.8 Summary . . . . . 9 - 15

10 Multi-state models...

10.1 Separating survival and movement . . . . . 10 - 4
10.2 A worked example: cost of breeding analysis . . . . . 10 - 6
10.3 States as 'groups' - multi-state models and the DM . . . . . .10 - 13
10.3.1 A simple metapopulation model - size, distance & quality . . . . . .10 - 17
10.4 Multi-state models as a unifying framework . . . . . .10 - 27
10.4.1 Simple example (1) – CJS mark-recapture as a MS problem . . . . . .10 - 28
10.4.2 Simple example (2) – dead-recovery analysis as a MS problem . . . . . .10 - 31
10.4.3 A more complex example - recruitment probability . . . . . .10 - 35
10.5 GOF testing and multi-state models . . . . . .10 - 46
10.5.1 Program U-CARE and GOF for time-dependent multi-state models . . . . . .10 - 46
10.5.2 MS models and the median $\hat{c}$ test . . . . . .10 - 48
10.6 multi-state models & unequal time intervals . . . . . .10 - 48
10.7 Summary . . . . . .10 - 49

11 Individual covariates

11.1 ML estimation and individual covariates . . . . . 11 - 2
11.2 Example 1 - normalizing selection on body weight . . . . . 11 - 3
11.2.1 Specifying covariate data in **MARK** . . . . . 11 - 5
11.2.2 Executing the analysis . . . . . 11 - 6
11.3 A more complex example - time variation . . . . . .11 - 14
11.4 The DM & individual covariates - some elaborations . . . . . .11 - 18
11.5 Visualizing the functional relationship with individual covariates . . . . . .11 - 23
11.6 Missing covariate values, time-varying covariates, and other complications... . . . . .11 - 29
11.6.1 Continuous individual covariates & multi-state models: a worked example . . . . . .11 - 30
11.7 Individual covariates as 'group' variables: a useful short-cut? . . . . . .11 - 36
11.7.1 Individual covariates for a binary classification variable . . . . . .11 - 36
11.7.2 Individual covariates for non-binary classification variables . . . . . .11 - 42
11.8 Model averaging and individual covariates . . . . . .11 - 44
11.9 GOF Testing and individual covariates . . . . . .11 - 46
11.10 Summary . . . . . .11 - 47

12 Jolly-Seber models in MARK

12.1 Protocol . . . . . 12 - 1
12.2 Data . . . . . 12 - 2
12.3 Multiple formulations of the same process . . . . . 12 - 3
12.3.1 The Original Jolly-Seber formulation . . . . . 12 - 4
12.3.2 *POPAN* formulation . . . . . 12 - 5
12.3.3 Link-Barker and Pradel-recruitment formulations . . . . . 12 - 7
12.3.4 Burnham JS and Pradel-λ formulations . . . . . 12 - 9
12.3.5 Choosing among the formulations . . . . . .12 - 11
12.3.6 Interesting tidbits . . . . . .12 - 13
12.4 Example 1 - estimating the number of spawning salmon . . . . . .12 - 13
12.4.1 *POPAN* formulation . . . . . .12 - 15
12.4.2 Link-Barker and Pradel-recruitment formulations . . . . . .12 - 25

12.4.3 Burnham Jolly-Seber and Pradel-λ formulations . . . . . 12 - 30
12.5 Example 2 - Muir's (1957) female capsid data . . . . . 12 - 36
12.5.1 POPAN formulation . . . . . 12 - 38
12.5.2 Link-Barker and Pradel-recruitment formulations . . . . . 12 - 43
12.5.3 Burnham Jolly-Seber and Pradel-λ formulations . . . . . 12 - 46
12.6 Final words . . . . . 12 - 50

13 Pradel models: recruitment, survival and population growth rate
13.1 Population growth: realized vs. projected . . . . . 13 - 1
13.2 Estimating realized λ . . . . . 13 - 2
13.2.1 Reversing encounter histories: φ and γ . . . . . 13 - 3
13.2.2 Putting it together: deriving λ . . . . . 13 - 4
13.3 Population λ versus Pradel's λ: are they equivalent? . . . . . 13 - 6
13.4 Pradel models in **MARK** . . . . . 13 - 6
13.5 an extension using the S and f parametrization... . . . . 13 - 14
13.6 Pradel models and Jolly-Seber estimation . . . . . 13 - 18
13.7 Summary . . . . . 13 - 18

14 Closed population capture-recapture models
14.1 The basic idea . . . . . 14 - 1
14.1.1 The Lincoln-Petersen estimator - a quick review . . . . . 14 - 2
14.2 Likelihood . . . . . 14 - 3
14.2.1 full likelihood approach . . . . . 14 - 3
14.2.2 conditional likelihood . . . . . 14 - 4
14.3 Model types . . . . . 14 - 6
14.3.1 Constraining the final p . . . . . 14 - 8
14.4 Encounter histories format . . . . . 14 - 9
14.5 Building models . . . . . 14 - 10
14.6 Closed population models and the design matrix . . . . . 14 - 14
14.7 Heterogeneity models . . . . . 14 - 20
14.7.1 Interpreting π . . . . . 14 - 25
14.8 Misidentification models . . . . . 14 - 26
14.8.1 joint heterogeneity and misidentification models . . . . . 14 - 27
14.9 Goodness-of-fit . . . . . 14 - 27
14.10 Model averaging and closed models . . . . . 14 - 27
14.10.1 Estimating CI for model averaged abundance estimates . . . . . 14 - 35
14.11 Parameter estimability in closed models . . . . . 14 - 40
14.12 Other applications . . . . . 14 - 41
14.13 Summary . . . . . 14 - 41
14.14 References . . . . . 14 - 41

15 The 'robust design'
15.1 Decomposing the probability of subsequent encounter . . . . . 15 - 1
15.2 Estimating γ: the classical 'live encounter' RD . . . . . 15 - 4
15.3 The RD extended – temporary emigration: γ' and γ'' . . . . . 15 - 6
15.3.1 γ parameters and multi-state notation . . . . . 15 - 7
15.3.2 illustrating the extended model: encounter histories & probability expressions . . . . . 15 - 8

15.3.3 Random (classical) versus Markovian temporary emigration . . . . . 15 - 9
15.3.4 Alternate movement models: no movement, and 'even flow' . . . . . .15 - 11
15.4 Advantages of the RD . . . . . .15 - 12
15.5 Assumptions of analysis under the RD . . . . . .15 - 13
15.6 RD (closed) in **MARK** – some worked examples . . . . . .15 - 13
15.6.1 Closed robust design – simple worked example . . . . . .15 - 13
15.6.2 Closed robust design – more complex worked example . . . . . .15 - 23
15.7 The multi-state closed RD . . . . . .15 - 29
15.7.1 multi-state closed RD – simple worked example . . . . . .15 - 29
15.8 The 'open' robust design... . . . . .15 - 39
15.8.1 Background . . . . . .15 - 39
15.8.2 The General ORD Model . . . . . .15 - 40
15.8.3 Implementing the ORD in **MARK**: simple example . . . . . .15 - 41
15.8.4 Dealing with unobservable states . . . . . .15 - 45
15.8.5 Which parameters can be estimated? . . . . . .15 - 46
15.8.6 Goodness of Fit . . . . . .15 - 47
15.8.7 Derived parameters from information within primary periods . . . . . .15 - 47
15.8.8 Analyzing data for just one primary period . . . . . .15 - 48
15.9 The robust design & unequal time intervals . . . . . .15 - 49
15.10 Literature . . . . . .15 - 50

16 Known-fate models
16.1 The Kaplan-Meier Method . . . . . 16 - 1
16.2 The Binomial Model . . . . . 16 - 3
16.3 Encounter Histories . . . . . 16 - 5
16.4 Worked example: black duck survival . . . . . 16 - 7
16.5 Pollock's staggered entry design . . . . . .16 - 11
16.5.1 Staggered entry – worked example . . . . . .16 - 11
16.6 Known fate and joint live-dead encounter models . . . . . .16 - 23
16.6.1 Live-dead and known fate models (1) 'radio impact' . . . . . .16 - 25
16.6.2 Live-dead and known fate models: (2) 'temporary emigration' . . . . . .16 - 25
16.7 Censoring . . . . . .16 - 25
16.8 Goodness of fit and known fate models . . . . . .16 - 26
16.9 Known-fate models and derived parameters . . . . . .16 - 27
16.10 Known-fate analyses and 'nest success models' . . . . . .16 - 27
16.11 Summary . . . . . .16 - 27

17 Nest survival models
17.1 Competing models of daily survival rate . . . . . 17 - 3
17.2 Encounter histories format . . . . . 17 - 4
17.3 Nest survival, encounter histories, & cell probabilities . . . . . 17 - 7
17.4 Building models . . . . . 17 - 8
17.4.1 Models that consider observer effects on DSR . . . . . .17 - 16
17.5 Model results . . . . . .17 - 17
17.6 Individual covariates and design matrix functions . . . . . .17 - 18
17.7 Additional applications of the nest success model . . . . . .17 - 18
17.8 Goodness of fit and nest survival . . . . . .17 - 19
17.9 Summary . . . . . .17 - 19

18 Mark-resight models
18.1 What is mark-resight? . . . 18 - 2
18.2 The mixed logit-normal mark-resight model . . . 18 - 5
18.2.1 No individually identifiable marks . . . 18 - 5
18.2.2 Individually identifiable marks . . . 18 - 8
18.3 The immigration-emigration mixed logit-normal model . . . 18 - 12
18.3.1 No individually identifiable marks . . . 18 - 13
18.3.2 Individually identifiable marks . . . 18 - 16
18.4 The Poisson-log normal mark-resight model . . . 18 - 19
18.4.1 Closed resightings only . . . 18 - 21
18.4.2 Full-likelihood robust design . . . 18 - 28
18.5 Which mark-resight model? Decision table . . . 18 - 34
18.6 Suggestions for mark-resight analyses in MARK . . . 18 - 34
18.7 References . . . 18 - 35

19 Young survival from marked adults
19.1 Assumptions . . . 19 - 2
19.2 Data . . . 19 - 2
19.3 Model Implementation . . . 19 - 3
19.4 Parameters and Sample Size . . . 19 - 5
19.5 Relationship with CJS and Multi-state Models . . . 19 - 6
19.6 Summary . . . 19 - 7

20 Density estimation...
20.1 Likelihood . . . 20 - 5
20.2 Implementation in MARK . . . 20 - 5
20.2.1 Estimate proportion on site or use the data? . . . 20 - 12
20.2.2 Threshhold Model . . . 20 - 14
20.3 Confidence Intervals . . . 20 - 16
20.4 Assumptions . . . 20 - 16
20.5 Summary . . . 20 - 17
20.6 References . . . 20 - 18

A Simulations in MARK . . .
A.1 Simulating CJS data . . . A - 1
A.1.1 Simulating CJS data - **MARK** simulations . . . A - 2
A.1.2 Simulating CJS data - **RELEASE** simulations . . . A - 10
A.2 Generating encounter histories - program **MARK** . . . A - 12
A.3 simulating data from a prior MARK analysis . . . A - 14
A.4 Simulation of robust design + closed capture data - special considerations . . . A - 18
A.5 Summary . . . A - 18

B The 'Delta method' . . .
B.1 Mean and variance of random variables . . . B - 1
B.2 Transformations of random variables and the Delta method . . . B - 3
B.3 Transformations of one variable . . . B - 7
B.3.1 A potential complication - violation of assumptions . . . B - 7

B.4 Transformations of two or more variables . . . B - 10
B.5 Delta method and model averaging . . . B - 27
B.6 Summary . . . B - 29

C RMark - an alternative approach to building linear models in MARK
C.1 RMark Installation and First Steps . . . C - 3
C.2 A simple example (return of the dippers) . . . C - 5
C.3 How RMark works . . . C - 9
C.4 Dissecting the function "mark" . . . C - 15
C.4.1 Function process.data . . . C - 15
C.4.2 Function make.design.data . . . C - 17
C.5 More simple examples . . . C - 21
C.6 Design covariates in **RMark** . . . C - 26
C.7 Comparing results from multiple models . . . C - 31
C.8 Producing model-averaged parameter estimates . . . C - 33
C.9 Quasi-likelihood adjustment . . . C - 34
C.10 Coping with identifiability . . . C - 35
C.11 Fixing real parameter values . . . C - 40
C.12 Data Structure and Import for **RMark** . . . C - 46
C.13 A more organized approach . . . C - 52
C.14 Defining groups with more than one variable . . . C - 56
C.15 More complex examples . . . C - 59
C.16 Individual covariates . . . C - 73
C.17 Multi-strata example . . . C - 80
C.18 Nest survival example . . . C - 88
C.19 Occupancy examples . . . C - 92
C.20 Known fate example . . . C - 97
C.21 Exporting to **MARK** interface . . . C.- 100
C.22 Using R for further computation and graphics . . . C.- 100
C.23 Problems and errors . . . C.- 103
C.24 A brief **R** primer . . . C.- 106

D Variance components and random effects models in MARK . . .
D.1 variance components - some basic background theory . . . D - 3
D.2 variance components estimation – worked examples . . . D - 7
D.2.1 Binomial survival – simple mean (no sampling covariance) . . . D - 7
D.2.2 Binomial example extended – simple trend . . . D - 12
D.2.3 what about sampling covariance? . . . D - 15
D.3 random effects models and shrinkage estimates . . . D - 17
D.3.1 the basic ideas... . . . D - 19
D.3.2 some technical background... . . . D - 21
D.3.3 Deriving an AIC for the random effects model . . . D - 23
D.4 random effects models – some worked examples . . . D - 24
D.4.1 binomial survival revisited – basic mechanics . . . D - 24
D.4.2 a more complex example – California mallard recovery data . . . D - 28
D.4.3 random effects – environmental covariates . . . D - 33
D.4.4 worked example – λ – Pradel model . . . D - 37
D.5 Model averaging? . . . D - 40

D.6 caveats, warnings, and general recommendations . . . . . D - 42
D.7 Summary . . . . . D - 44
D.8 Literature . . . . . D - 45

E Markov Chain Monte Carlo (MCMC) estimation in MARK . . .
E.1 Variance components analysis revisited - MCMC approach . . . . . E - 5
E.1.1 Example 1 - binomial survival re-visited . . . . . E - 5
E.1.2 estimating the hyperparameters μ and σ . . . . . E - 18
E.1.3 Example 2 – California mallard survival re-visited . . . . . E - 23
E.1.4 Example 3 - environmental covariates re-visited . . . . . E - 26
E.2 Hyperdistributions between structural parameters . . . . . E - 35
E.2.1 Example 1 - dead recovery analysis (corr{**S**, **f**}) . . . . . E - 36
E.2.2 Example 2 - Pradel model . . . . . E - 43
E.3 caveats, warnings, and general recommendations . . . . . E - 46
E.4 Summary . . . . . E - 48
E.5 Literature . . . . . E - 48

F Parameter identifiability by data cloning...
F.1 worked example (1) – the Dippers . . . . . F - 2
F.1.1 structural identifiability – confounded parameters . . . . . F - 3
F.1.2 'boundary problems' – data limits and link functions . . . . . F - 5
F.1.3 choice of link function – does it matter? . . . . . F - 8
F.2 worked example (2) – AFS monograph example . . . . . F - 10
F.3 worked example (3) – robust design example . . . . . F - 10
F.4 worked example (4) – Pradel model example . . . . . F - 13
F.5 Limitations & other thoughts and approaches . . . . . F - 17
F.6 Summary . . . . . F - 21
F.7 Literature Cited . . . . . F - 21

G The 'data bootstrap. . .'
G.1 Empirical example . . . . . G - 2
G.2 Extensions . . . . . G - 10
G.3 Limitations . . . . . G - 11
G.4 Literature Cited . . . . . G - 12

Program MARK – a 'gentle introduction'

Program **MARK** is the most comprehensive and widely used software application currently available for the 'analysis of data from *marked* individuals' (hence the name **MARK**). **MARK** is a very flexible and powerful program, with many options, and a lot of technical and theoretical sophistication. It encompasses virtually all currently used methods for analysis of marked individuals - including many new approaches only recently described in the primary literature.

As such, **MARK** is not a program that you can learn to use without some instruction. Unfortunately, the only documentation for **MARK** consists entirely of the Windows 'help file' which accompanies **MARK**. This is not to slight the help file – it is extremely comprehensive, and covers almost all of the 'hard details' you might want to know. For people with a strong background in analysis of these sorts of data, and especially experienced users of **E/M-SURGE** and **POPAN** (the other 'big' applications in common use), the help file alone may, in fact, be sufficient to get you 'up and running' with **MARK** with only a bit of work. **MARK** draws heavily on the strengths of other applications, and aspects and underlying principles are (to varying degrees) similar across many applications.

However, for the 'new user', who may have little to no background in the analysis of these sort of data, learning how to use **MARK** from the help file alone is very inefficient, and is often a frustrating exercise. This type of user needs a different type of 'documentation'. It was with this type of user in mind that we developed this book - a comprehensive example-driven 'tutorial' on the theory, mechanics and practice of using **MARK**.

Of course, **MARK** is not the only program available for analysis of encounter data from marked individuals (see `http://www.phidot.org/software/` for pointers to other available software), so you may wonder "why bother with **MARK**?". The short answer is that **MARK** offers far more flexibility and power in statistical modeling and hypothesis testing than other widely available and frequently used programs. It also uses a consistent, and familiar 'Windows interface', and allows the user to work with a consistent data format throughout. If you're just starting out, and have to pick one program to become proficient with, we strongly suggest you spend your time with **MARK**. Of course, there may be reasons why you don't want to use **MARK**, but on average, it'll be well worth your while.

About this book

This book is intended to allow you to (in effect) 'teach yourself how to use **MARK**'. We have included much of the material we normally cover in the classroom or during workshops, placing as much emphasis on "why things work the way they do" as on "now…press this button". Our basic view of learning to use software is that the only way to really master an application is to understand what it is doing, and then to practice the mechanics of the application (over and over again).

Having said that, it is worth letting you know right from the beginning that this is **not** a book on the theory of analysis of data from marked individuals (in the strict sense), and should not be cited as such.* This guide is intended simply to be an accessible means by which you can learn *how* to use **MARK**. In the process, however, we do cover a fair bit of the 'conceptual theory'. If you're an experienced analyst of these sort of data, you'll quickly find which parts you can skip, and which you can't. Regardless, we urge you to read the current literature (see below) – it is the only way to keep up with the many recent developments in the analysis of data from marked individuals.

Structure of the book

Chapters 1 through 7 are the 'core' mechanical and conceptual 'skill-building' chapters. Chapters 8 and higher are focused on more advanced applications. For newcomers, we **strongly** suggest that you work through Chapters 1 through 7 first. And, by 'working through', we mean sitting at the computer, with this book, and working through **all** of the computer exercises. This book is largely based on the premise that you 'learn by doing'. Chapter 1 provides a simple introduction to some of the ideas and theory. Chapter 2 covers the basics of data formatting (the obvious first step to analyzing your data). Chapters 3 to 7 provide detailed instruction on the 'basics' of using **MARK**, within the context of 'standard' open population mark-recapture analysis. We decided to begin with basic mark-recapture for two reasons. First, it is the basis for most of the commonly used software applications currently in wide use. Since most experienced analysts will probably have some level of experience with one or more of these applications, building the core introduction around mark-recapture seems to present the minimal learning curve. Second, and perhaps more importantly for beginners, if you understand basic mark-recapture analysis, you can pick up the other types of analysis fairly quickly.

In these first chapters, we will take you, step-by-step, through the process of using **MARK**, working for the most part with 'practice' data sets†, starting with the basic rudiments, and ending with some fairly sophisticated examples. Our goal is to provide you with enough understanding of how **MARK** works so that even if we don't explicitly cover the particular problem you're working on, you should be able to figure out how to approach the problem with **MARK**, on your own. In fact, we measure the success of this book by how little you'll need to refer to it again, once you've gone through all of the core chapters. Once you have worked through Chapters 1 through 7, you should be able to jump to any of the following chapters with relative ease. All succeeding chapters are reasonably self-contained, but do presume you're familiar with basic mark-recapture theory, and (especially) how it is applied in **MARK**.

begin sidebar

sidebars - extra information

Interspersed throughout most chapters will be '`sidebars`' - small snippets of technical information, conceptual arm-waving, or other information which we think is potentially useful for you to read – but not so essential that they can't be skipped over to maintain your flow of the reading of the main body of the text. Whenever you come across one of these `sidebar` items, it might be worth at least reading the first few lines to see what it refers to.

end sidebar

* We're occasionally asked how to properly cite this book. Easy answer – please don't. This book is not a 'technical reference', but a 'software manual'. The various 'technical' bits in the book (i.e., suggestions on how to approach some sorts of analysis, guides to interpreting results...) are drawn from the primary literature, which should be cited in all cases.

† Most of the practice data sets are contained in the file `markdata.zip`, which can be downloaded from the same website you accessed to download this book. The last item of the drop-down menu where you select chapters (left-hand side of the page) is a link to the example files. If you can't find the files there, check the `/mark/examples` subdirectory that is created when you install **MARK**.

Getting MARK and installing it on your computer

The primary source for program **MARK** is Gary White's **MARK** web page, which is currently located at

`http://www.cnr.colostate.edu/~gwhite/mark/mark.htm`

New versions, miscellaneous notes, and general comments concerning **MARK** are found there, as well as links to lecture notes, and other relevant information. And, if you're reading this, then you've obviously found the 'other **MARK** website', maintained by Evan Cooch.

`http://www.phidot.org/software/mark/`

The purpose of this 'other' web site is twofold: (i) to provide access to this book, and (ii) to provide a locally mirrored copy of the **MARK** install files (either one website or the other is always likely to be 'up and running'). Updates to **MARK** are relatively frequent. At present, the only way to check for new versions is to periodically visit either of these two websites, and look to see if another version has been posted. Alternatively, you can register for the online **MARK** discussion forum (see below), which will send periodic emails announcing new releases of **MARK**.

Now, about installing **MARK**. Before we go into the details - some quick comments:

- **MARK** is a Windows program, and is intended to be run under a Windows 'environment'. For most users, these means a machine running Windows XP → Windows 7/8. However, as the technology underlying 'Windows emulation' and 'virtualization' software gets better and better, it has become more and more tractable to run **MARK** on a non-Windows platform (e.g., using VMWare, VirtualBox or wine under Linux, or Parallels, Bootcamp or SoftPC on a Mac). Some details about running **MARK** on a non-Windows machine can be found on the **MARK** website.
- Running **MARK** also requires a 'real computer' - we recommend *minimally* a machine with a CPU clocked at 2 GHz or better (**MARK** supports and makes use of multi-core processors), with at least 2 GB of RAM (>2 GB strongly recommended). We also suggest getting a decent sized monitor - no less than 19 inches (you'll discover why we make this recommendation the first time you pull up a 'big, ugly' design matrix).

You install **MARK** using a fairly standard setup program. Once you've downloaded the **MARK** `setup.exe` program, simply double-click it, and off you go! It is a fairly standard Windows installer, with prompts for where you want to install **MARK**, and so forth. The installation is generally uneventful. Once the install program has finished, you're done. The install routine should have placed a short-cut to **MARK** for you on your desktop.

begin sidebar

upgrading from an earlier installation

As noted, updates to **MARK** are fairly frequent (with the pace of change being roughly proportional to the rate at which new methods enter the literature). To upgrade an existing **MARK** installation, you should

1. **uninstall the old version**, using the 'uninstall software' option from the Windows control panel (ignore any error messages you might get about Windows not being able to unregister certain items - these are spurious). If you really want to be thorough, follow this by manually deleting the **MARK** subdirectory as well (although this isn't really necessary).

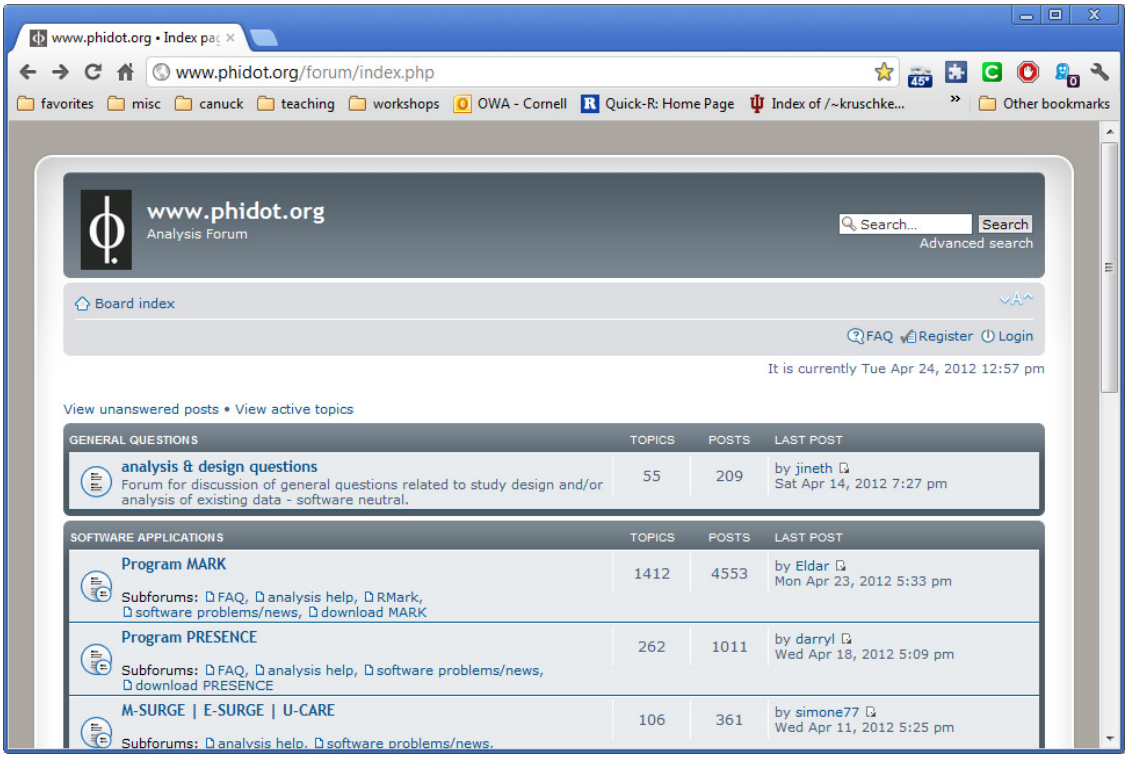
2. **install the new version**. For some operating systems, you *may* get an error message or two concerning problems trying to register certain graphics components - ignore these.
3. **test the installation**. double-click the **MARK** icon, and make sure it starts up correctly. If it does, you should be fine.

end sidebar

Finding help

No matter how good the documentation, there will always be things that remain unclear, or simply aren't covered in this book (although we keep trying). As such, it's nice to have some options for getting help (beyond the earlier admonition to 'check the help file'). As such, we have created a web-based discussion forum for just this purpose - a place where you can ask questions, make suggestions for **MARK**[*], and so forth. The forum can be accessed at

`http://www.phidot.org/forum`



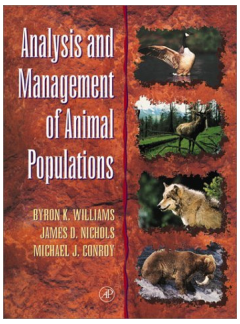
In addition to providing a resource for getting answers to specific technical questions, registering for the forum[†] is also a convenient way to learn about recent changes to **MARK** (and this book), and finding out about upcoming workshops and training sessions.

[*] The forum also hosts similar discussions for a number of other software applications; e.g., **PRESENCE**, **M/E-SURGE**...
[†] Registration for the forum is free, and you have a fair bit of control over how much 'email traffic' it generates.

References & background reading

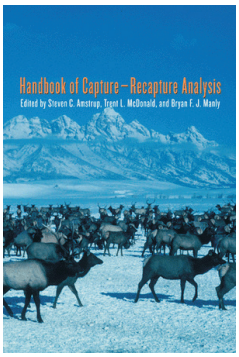
The literature for analysis of data from marked individuals is very large - and growing at an exponential rate (100-150 new papers per year in recent years). As such, it's easy to feel that keeping up with the literature is not even remotely tractable. Don't fret - it's unlikely anyone reads all the papers.*

Fortunately, there have been several recently published books, which do much of the collation and synthesis of this large literature for you - we **strongly** suggest that you get access (in some fashion) to the following 3 books:



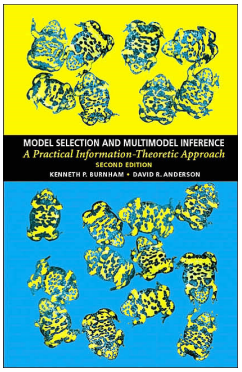
Analysis and Management of Animal Populations - Ken Williams, Jim Nichols, and Mike Conroy. (2002) Academic Press. 1040 pages.

A staggering volume that is the *de facto* standard reference for the integration of modeling, estimation, and management, written by 3 of the luminaries in the field. It provides a superb synthesis of most of the vast literature on estimation from data from marked individuals.



Handbook of Capture-Recapture Analysis - Steve Amstrup, Lyman MacDonald, and Bryan Manly. (2006) Princeton University Press. 296 pages.

In some ways, a precis of some of the key 'estimation' sections of the WNC book (above), in others, a more detailed 'guide' to several extensions to methods discussed in WNC. A very good, compact summary of estimation methods, with a focus on practical application.



Model Selection and Multi-Model Inference (2nd Edition) - Ken Burnham and David Anderson. (2002) Spring-Verlag. 496 pages.

So you want to fit models to data, eh? Well, fundamental to this process is the issue of selecting amongst such models. How should you do this? Burnham and Anderson cover this critical issue in great detail - and in so doing, will give you a solid basis for the mechanics, and theory, of model selection as applied to analysis of data from marked individuals.

Collectively, these books represent the *minimum* library you should have at your disposal, and are essential companions to this book.

* With the exception of Jim Nichols, who is a 'special case' in several respects...

Acknowledgements

The first draft of this book was written in 1998. It was approximately 150 pages in length. It is now >1,000 pages, and has gone through numerous revisions, based almost entirely on the comments, corrections and feedback submitted by many of the several thousand people who have used the book in its various incarnations. In addition, several new chapters have recently been contributed by some of our colleagues. These contributions are so significant that we now consider ourselves as merely 'editors' of the larger effort by the community of **MARK** users to document the software. Any strengths of this book come from these collegial interactions – its failings, however, are our fault alone. We believe in earnest that the only truly 'dumb' question is one never asked. Hopefully, most of your questions concerning the use of program **MARK** are answered here.

Ithaca, New York EGC
Ft. Collins, Colorado GCW

2014

This publication may not be reproduced, stored, or transmitted in any form except for (i) fair use for the purposes of research, teaching (which includes printing for instructional purposes) or private study, or for criticism or review, or (ii) with the express written permission of the editors, or authors of individual chapters. Permission is summarily granted for use governed by these terms. Enquiries concerning reproduction outside these terms should be directed to the editors.

CHAPTER 1

First steps. . .

We'll introduce the basic idea for analysis of data from marked individuals by means of a simple example. Suppose you are interested in exploring the costs of reproduction on survival of some species of your favorite taxa (say, a species of bird). The basic idea is pretty simple: an individual that spends a greater proportion of available energy on breeding may have less available for other activities which may be important for survival. In this case, individuals putting more effort into breeding (i.e., producing more offspring) may have lower survival than individuals putting less effort into breeding. On the other hand, it might be that individuals that are of better 'quality' are able to produce more offspring, such that there is no relationship between 'effort' and survival. You decide to reduce the confounding effects of the 'quality' hypothesis by doing an experiment. You take a sample of individuals who all produce the same number of offspring (the idea being, perhaps, that if they had the same number of offspring in a particular breeding attempt, that they are likely to be of similar quality). For some of these individuals, you increase their 'effort' by adding some offspring to the nest (i.e., more mouths to feed, more effort expended feeding them). For others, you reduce effort by removing some offspring from the nest (i.e., fewer mouths to feed, less effort spent feeding them). Finally, for some individuals, you do not change the number of offspring, thus creating a control group.

As described, you've set up an 'experiment', consisting of a control group (unmanipulated nests), and 2 treatment groups: one where the number of offspring has been reduced, and one where the number of offspring has been increased. For convenience, call the group where the number of offspring was increased the 'addition' group, and call the group where the number of offspring was reduced the 'subtraction' group. Your hypothesis might be that the survival probability of the females in the 'addition' group should be lower than the control (since the females with enlarged broods might have to work harder, potentially at the expense of survival), whereas the survival probability of the females in the 'subtraction' group should be higher than the control group (since the females with reduced broods might not have to work as hard as the control group, potentially increasing their survival). To test this hypothesis, you want to estimate the survival of the females in each of the 3 groups. To do this, you capture and individually mark the adult females at each nest included in each of the treatment groups (control, additions, subtractions). You release them, and come back at some time in future to see how many of these marked individuals are 'alive' (the word 'alive' is written parenthetically for a reason which will be obvious in a moment).

Suppose at the start of your study (time t) you capture and mark 50 individuals in each of the 3 groups. Then, at some later time (time t+1), you go back out in the field and encounter alive 30 of the marked individuals from the 'additions' treatment, 35 of the marked individuals from the control group, and 30 individuals from the 'subtractions' treatment. The 'encounter data' from our study are tabulated at the top of the next page.

© Cooch & White (2014)

group	t	$t+1$
additions	50	30
control	50	38
subtractions	50	30

Hmm. This seems strange. While you predicted that the 2 treatment groups would differ from the controls, you did not predict that the results from the two treatments would be the same. What do these results indicate? Well, of course, you could resort to the time-honored tradition of trying to concoct a parsimonious 'post-hoc adaptationist' story to try to demonstrate that (in fact) these results 'made perfect sense', according to some 'new twist to underlying theory'. However, there is another possibility - namely, that the analysis has not been thoroughly understood, and as such, interpretation of the results collected so far needs to be approached very cautiously.

1.1. Return 'rates'

Let's step back for a moment and think carefully about our experiment - particularly, the analysis of 'survival'. In our study, we marked a sample of individual females, and simply counted the numbers of those females that were subsequently seen again on the next sampling occasion. The implicit assumption is that by comparing relative proportions of 'survivors' in our samples (perhaps using a simple χ^2 test), we will be testing for differences in 'survival probability'. However (and this is the key step), is this a valid assumption? Our data consist of the number of marked and released individuals that were encountered again at the second sampling occasion. While it is obvious that in order to be seen on the second occasion, the marked individual must have survived, is there anything else that must happen? The answer (perhaps obviously, but in case it isn't) is 'yes' - the number of individuals encountered on the second sampling occasion is a function of 2 probabilities: the probability of survival, and the probability that conditional on surviving, that the surviving individual is encountered. While the first of these 2 probabilities is obvious (and is in fact what we're interested in), the second may not be. This second probability (which we refer to generically as the 'encounter probability') is the probability that given that the individual is alive and in the sample, that it is in fact encountered (e.g., seen, or 'visually encountered'). In other words, simply because an individual is alive and in the sampling area may not guarantee that it is encountered. So, the proportion of individuals that were encountered alive on the second sampling occasion (which is often referred to in the literature as 'return rate'*) is the product of 2 different probability processes: the probability of surviving and returning to the sampling area (which we'll call 'apparent' or 'local' survival), and the probability of being encountered, conditional on being alive an in the sample (which we'll call 'encounter probability'). So, 'return rate' = 'survival probability' × 'encounter probability'. Let's let φ (pronounced '*fee*' or '*fie*', depending on where you come from) represent the 'local survival probability', and p represent the 'encounter probability'. Thus, we would write 'return rate' = φp.

So, why do we care? We care because this complicates the interpretation of 'return rates' - in our example, differences in 'return rates' could reflect differences in the probability of survival, or they could reflect differences in encounter probability, or both! Similarly, lack of differences in 'return rates' (as we see when comparing the 'additions' and 'subtractions' treatment groups in our example) may not indicate 'no differences in survival' (as one interpretation) – there may in fact be differences in survival, but corresponding differences in encounter probability, such that their products ('return

* The term 'return rate' is something of a misnomer, since it is not a *rate*, but rather a *proportion*. However, because the term 'return rate' is in wide use in the literature, we will continue to use it here.

rate') are equal. For example, in our example study, the 'return rate' for both the 'additions' and 'subtractions' treatment groups is the same: $(30/50) = 0.6$. Our initial 'reaction' might have been that these data did not support our hypothesis predicting difference in survival between the 2 groups. However, suppose that in fact the 'treatment' (i.e., manipulating the number of offspring in the nest) not only influenced survival probability (as was our original hypothesis), but also potentially influenced encounter probabilities? For example, suppose the true survival probability of the 'additions' group was $\varphi_{add} = 0.65$ (i.e., a 65% chance of surviving from t to t+1), while for the 'subtractions' group, the survival probability is $\varphi_{sub} = 0.80$ (i.e., an 80% chance of surviving). However, in addition, suppose that the encounter probability for the 'additions' group was $p_{add} = 0.923$ (i.e., a 92.3% chance that a marked individual will be encountered, conditional on it being alive and in the sampling area), while for the 'subtractions' group, the encounter probability was $p_{sub} = 0.75$ (we'll leave it to proponents of the adaptationist paradigm to come up with a 'plausible' explanation for such differences). While there are clear differences between the 2 groups, the products of the 2 probabilities are the same: $(0.65 \times 0.923) = 0.6$, and $(0.8 \times 0.75) = 0.6$. In other words, it is difficult to compare 'return rates', since differences (or lack thereof) could reflect differences or similarities in the 2 underlying probabilities (survival probability, and encounter probability).

1.2. A more robust approach

How do we solve this dilemma? Well, the solution we're going to focus on here (and essentially for the next 900 pages or so) is to collect more data, and using these data, separately estimate all of the probabilities (at least, when possible) underlying the encounters of marked individuals. Suppose for example, we collected more data for our experiment, on a third sampling occasion (at time $t + 2$). On the third occasion, we encounter individuals marked on the first occasion. But, perhaps some of those individuals encountered on the third occasion were not encountered on the second occasion. How would we be able to use these data? First, we introduce a simple bookkeeping device, to help us keep track of our 'encounter' data (in fact, we will use this bookkeeping system throughout the rest of the book – discussed in much more detail in Chapter 2). We will 'keep track' of our data using what we call '*encounter histories*'. Let a '1' represent an encounter with a marked individual (in this example, we're focusing only on 'live encounters'), and let a '0' indicate that a particular marked individual was not seen on a particular occasion. Now, recall from our previous discussion that a '0' could indicate that the individual had in fact died, but it could also indicate that the individual was in fact still alive, but simply not encountered (the problem we face, as discussed, is how to differentiate between the two possibilities). For our 3 occasion study, where individuals were uniquely marked on the first occasion only, there are 4 possible encounter histories:

encounter history	*interpretation*
111	captured and marked on the first occasion, alive and encountered on the second occasion, alive and encountered on the third occasion
110	captured and marked on the first occasion, alive and encountered on the second occasion, and either (i) dead by the third occasion, or (ii) alive on the third occasion, but not encountered
101	captured and marked on the first occasion, alive and not encountered on the second occasion, and alive and encountered on the third occasion

`100`	captured and marked on the first occasion, and either (i) dead by the second occasion, (ii) alive on the second occasion, and not encountered, and alive on the third occasion and not encountered, (iii) alive on the second occasion, and not encountered, and dead by the third occasion

You might be puzzled by the verbal explanation of the third encounter history: `101`. How do we know that the individual is alive at the second occasion, if we didn't see it? Easy - we come to this conclusion logically, since we saw it alive at the third occasion. And, if it was alive at occasion 3, then it must also have been alive at occasion 2. But, we didn't see it on occasion 2, even though we know (logically) that it was alive. This, in fact, is one of the key pieces of logic – the individual was alive at the second occasion but not seen. If p is the probability of detecting (encountering) an individual given that it is alive and in the sample, then $(1 - p)$ is the probability of missing it (i.e., not detecting it). And clearly, for encounter history '`101`', we 'missed' the individual at the second occasion.

All we need to do next is take this basic idea, and formalize it. As written (above), you might see that each of these encounter histories could occur due to a specific sequence of events, each of which has a corresponding probability. Let φ_i be the probability of surviving from time (i) to (i+1), and let p_i be the probability of encounter at time (i). Again, if p_i is the probability of encounter at time (i), then $(1 - p_i)$ is the probability of not encountering the individual at time (i).

Thus, we can re-write the preceding table as:

encounter history	*probability of encounter history*
`111`	$\varphi_1 p_2 \varphi_2 p_3$
`110`	$\varphi_1 p_2 [\varphi_2 (1 - p_3) + (1 - \varphi_2)]$ $= \varphi_1 p_2 (1 - \varphi_2 p_3)$
`101`	$\varphi_1 \, (1 - p_2) \, \varphi_2 p_3$
`100`	$(1 - \varphi_1) + \varphi_1 (1 - p_2)(1 - \varphi_2) + \varphi_1 (1 - p_2) \varphi_2 (1 - p_3)$ $= 1 - \varphi_1 p_2 - \varphi_1 (1 - p_2) \varphi_2 p_3$

(If you don't immediately see how to derive the probability expressions corresponding to each encounter history, not to worry: we will cover the derivations in much more detail in later chapters).

So, for each of our 3 treatment groups, we simply count the number of individuals with a given encounter history. Then what? Once we have the number of individuals with a given encounter history, we use these frequencies to *estimate* the probabilities which give rise to the observed frequency. For example, suppose for the 'additions' group we had $N^{111} = 7$ (where N^{111} is the number of individuals in our sample with an encounter history of '111'), $N^{110} = 2$, $N^{101} = 5$, and $N^{100} = 36$. So, of the 50 individuals marked at occasion 1, only (7+2+5) = 14 individuals were subsequently encountered alive (at either sampling occasion 2, sampling occasion 3, or both), while 36 were never seen again. Suppose for the 'subtractions' group we had $N^{111} = 5$, $N^{110} = 7$, $N^{101} = 2$, and $N^{100} = 36$. Again, 14 total individuals encountered alive over the course of the study.

However, even though both treatment groups (additions and subtractions) have the same overall 3-year return rate (14/50 = 0.28), we see clearly that the frequencies of the various encounter histories differ between the groups. This indicates that there are differences among encounter occasions in

survival probability, or encounter probability (or both) between the 2 groups, despite no difference in overall return rate. The challenge, then, is how to estimate the various probabilities (parameters) in the probability expressions, and how to determine if these parameter estimates are different between the 2 treatment groups.

An *ad hoc* way of getting at this question involves comparing ratios of frequencies of different encounter ratios. For example,

$$\frac{N^{111}}{N^{101}} = \frac{\varphi_1 p_2 \varphi_2 p_3}{\varphi_1 (1-p_2) \varphi_2 p_3} = \frac{p_2}{1-p_2}$$

So, for the 'additions' group, $(N^{111}/N^{101}) = (7/5) = 1.4$. Thus, $\hat{p}_{(2,add)} = 0.583$. In contrast, for the 'subtractions' group, $(N^{111}/N^{101}) = (5/2) = 2.5$. Thus, $\hat{p}_{(2,sub)} = 0.714$. Once we have estimates of p_2, we can see how we could substitute these values into the various probability expressions to solve for some of the other parameter (probability) values. However, while this is reasonably straightforward (at least for this very simple example), what about the question of 'is this difference between the two different $\hat{p}_2$ values meaningful/significant?'. To get at this question, we clearly need something more - in particular we need to be able to come up with estimates of the uncertainty (variance) in our parameter estimates. To do this, we need a robust statistical tool.

1.3. Maximum likelihood theory - the basics

Fortunately, we have such a tool at our disposal. Analysis of data from marked individuals involves making inference concerning the probability structure underlying the sequence of events that we observe. Maximum likelihood (ML) estimation (courtesy of Sir Ronald Fisher) is the workhorse of analysis of such data. While it is possible to become fairly proficient at analysis of data from marked individuals without any real formal background in ML theory, in our experience at least a passing familiarity with the concepts is helpful. The remainder of this (short) introductory chapter is intended to provide a simple (very) overview of this topic. The standard 'formal' reference is the 1992 book by AWF Edwards ('*Likelihood*', Johns Hopkins University Press). Readers with significant backgrounds in the theory will want to skip this chapter, and will absolutely refrain from comment as to the necessary simplifications we make.

So here we go. . .the basics of maximum likelihood theory without (much) pain. . .

1.3.1. Why maximum likelihood?

The method of maximum likelihood provides estimators that are both reasonably intuitive (in most cases) and several have some 'nice properties' (at least statistically):

1. The method is very broadly applicable and is simple to apply.
2. Once a maximum-likelihood estimator is derived, the general theory of maximum-likelihood estimation provides standard errors, statistical tests, and other results useful for statistical inference. More technically:
   (a) maximum-likelihood estimators (MLE) are *consistent*.
   (b) they are asymptotically unbiased (although they may be biased in finite samples).
   (c) they are asymptotically *efficient* - no asymptotically unbiased estimator has a smaller asymptotic variance.

(d) they are asymptotically normally distributed - this is particularly useful since it provides the basis for a number of statistic 'tests' based on the normal distribution (discussed in more detail in Chapter 4).

(e) if there is a *sufficient statistic* for a parameter, then the MLE of the parameter is a function of a sufficient statistic.*

3. A disadvantage of the maximum likelihood method is that it frequently requires strong assumptions about the structure of the data.

1.3.2. Simple estimation example - the binomial coefficient

We will introduce the basic idea behind maximum likelihood (ML) estimation using a simple, and (hopefully) familiar example: a binomial model with data from a flip of a coin. Much of the analysis of data from marked individuals involves ML estimation of the probabilities defining the occurrence of one or more events. Probability events encountered in such analyses often involve binomial or multinomial distributions. As you might appreciate, there is a simple, logical connection between binomial probabilities, and analysis of data from marked individuals, since many of the fundamental parameters we are interested in are 'binary' (having 2 possible states). For example, survival probability (live or die), detection probability (seen or not seen), and so on. Like a coin toss (head or tail), the estimation methods used in the analysis of data from marked individuals are deeply rooted in basic binomial theory. Thus, a brief review of this subject is in order.

To understand binomial probabilities, you need to first understand *binomial coefficients*. Binomial coefficients are commonly used to calculate the number of ways (combinations) a sample size of n can be taken without replacement from a population of N individuals.

$$\binom{N}{n} = \frac{N!}{n!(N-n)!} \tag{1.1}$$

This is read as 'the number of ways (combination) a sample size of n can be taken (without replacement) from a population of size N'. Think of N as the number of organisms in a defined population, and let n be the sample size, for example. Recall that the '!' symbol means *factorial* (e.g., $5! = 5\times4\times3\times2\times1 = 120$).

A quick example – how many ways can a sample of size 2 (i.e., $n = 2$) be taken from a population of size 4 (i.e., $N = 4$)? Just to confirm we're getting the right answer, let's first derive the answer by 'brute force'. Let the individuals in the sample all have unique marks: call them individuals **A**, **B**, **C** and **D**, respectively. So, given that we sample 2 at a time, without replacement, the possible combinations we could draw from the 'population' are:

AB	AC	AD	BC	BD	CD
BA	CA	DA	CB	DB	DC

So, 6 total different combinations are possibly selected (6, not 12 – the pair in each column are equivalent; e.g., 'AB' and 'BA' are treated as equivalent).

* Sufficiency is the property possessed by a statistic, with respect to a parameter, when no other statistic which can be calculated from the same sample provides any additional information as to the value of the parameter. For example, the arithmetic mean is sufficient for the mean (μ) of a normal distribution with known variance. Once the sample mean is known, no further information about μ can be obtained from the sample itself.

So, does this match with $\binom{4}{2}$?

$$\binom{4}{2} = \frac{4!}{2!(4-2)!} = \frac{24}{2(2)} = \frac{24}{4} = 6$$

Nice when things work out, eh? OK, to continue - we use the binomial coefficient to calculate the *binomial probability*. For example, what is the probability of 5 heads in 20 tosses of a fair coin. Each individual coin flip is called a *Bernoulli trial*, and if the coin is fair, then the probability of getting a head is $p = 0.5$, while the probability of getting a tail is $(1 - p) = 0.5$ (commonly denoted as q). So, given a fair coin, and $p = q = 0.5$, then the probability of y heads in N flips of the coin is:

$$f(y \mid N, p) = \binom{N}{y} p^y (1-p)^{(N-y)} \tag{1.2}$$

The left-hand side of the equation is read as 'the probability of observing y events given that we do the experiment – toss the coin N times, and given that the probability of a head in any given experiment (i.e., toss of the coin) is p'. Given that $N = 20$, and $p = 0.5$, then the probability of getting exactly 5 heads in 20 tosses of the coin is:

$$f(5 \mid 20, p) = \binom{20}{5} p^5 (1-p)^{(20-5)}$$

First, we calculate $\binom{20}{5} = 15{,}504$ (note: 20! is a **huge** number). If $p = 0.5$, then $f(5 \mid 20, 0.5) = (15{,}504 \times 0.03125 \times 0.000030517578125) = 0.0148$. So, there is a 1.48% chance of having 5 heads out of 20 coin flips, if $p = 0.5$.

Now, in this example, we are assuming that we *know* both the number of times that we toss the coin, and (critically) the probability of a head in a single toss of the coin. However, if we are studying the survival of some organism, for example, what information on the left side of the probability equation (above) would we know? Well, hopefully we know the number of individuals marked (N). Would we know the survival probability (in the above, the survival probability would correspond to p - later, we'll call it S)? No! Clearly, this is what we're trying to estimate.

So, given the number of marked individuals (N) at the start of the study and the number of individuals that survive (y), how can we estimate the survival probability p? Easy enough, actually - we simply work 'backwards' (more or less). We find the value of p that maximizes the *likelihood* ($\mathcal{L}$) that we would observe the data we did.* So, for example, what would the value of p have to be to give us the observed data?

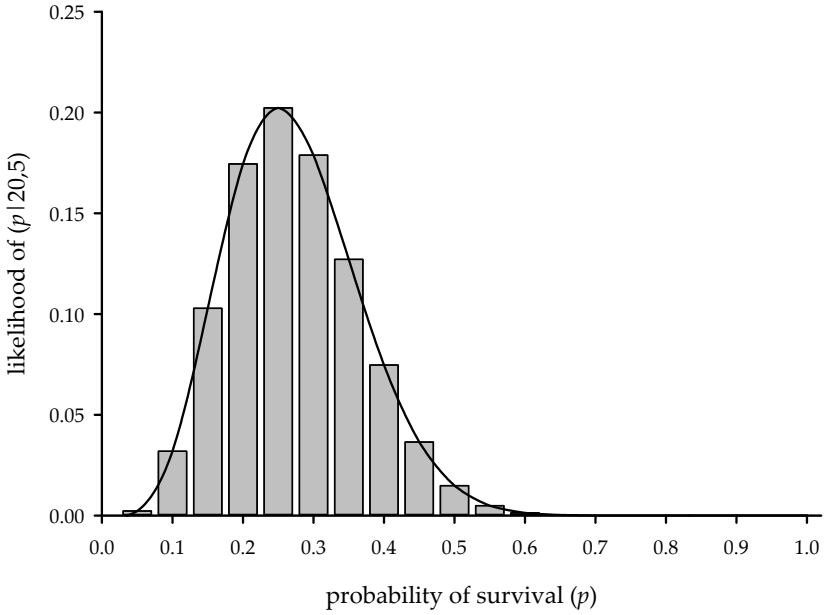
Formally, we write this as:

$$\mathcal{L}(p \mid N, y) = \binom{N}{y} p^y (1-p)^{(N-y)} \tag{1.3}$$

We notice that the right-hand side of eqn. (1.3) is identical to what it was before in eqn. (1.2) - but the left hand side is different in a subtle, but critical way. We read the left-hand side now as 'the likelihood $\mathcal{L}$ of survival probability p given that N individuals were released and that y survived'. Now, suppose $N = 20$, and that we see 5 individuals survive (i.e., $y = 5$). What would p have to be to maximize the chances of this occurring? We'll try a 'brute force' approach first, simply seeing what happens if we

* The word 'likelihood' is often used synonymously for 'probability' but in statistical usage, they are not equivalent. One may ask 'If I were to flip a fair coin 10 times, what is the *probability* of it landing heads-up every time?' or 'Given that I have flipped a coin 10 times and it has landed heads-up 10 times, what is the *likelihood* that the coin is fair?' but it would be improper to switch 'likelihood' and 'probability' in the two sentences.

set $p = 0, 0.1, 0.2, \ldots$, and so on. Look at the following plot of the binomial probability calculated for different values of p:



As you see, the probability of 'observing 5 survivals out of 20 individuals' rises to a maximum when p is 0.25. In other words, if p, which is unknown, were 0.25, then this would correspond to the maximal probability of observing the data of 5 survivors out of 20 released individuals. This graph shows that some values of the unknown parameter p are 'relatively unlikely' (i.e., those with low likelihoods), given the data observed. The value of the parameter p at which this graph is at a maximum is the most likely value of p (the probability of a head), given the data. In other words, the chances of actually observing 11 heads and 5 tails are maximal when p is at the maximum point of the curve, and the chances are less when you move away from this point.

While graphs are useful for getting a 'look' at the likelihood, we prefer a more elegant way to estimate the parameter. If you remember any of your basic calculus at all, you might recall that what we want to do is find the maximum point of the likelihood function. Recall that for any function $y = f(x)$, we can find the maximum inflection point over a given domain by setting the first derivative dy/dx to zero and solving. This is exactly what we want to do here, except that we have one preliminary step - we 'could' take the derivative of the likelihood function as written, but it is simpler to convert everything to logarithms first. The main reason to do this is because it simplifies the analytical side of things considerably. The log-transformed likelihood, now referred to as a 'log-likelihood', is denoted as $\ln \mathcal{L}\,(q \mid \text{data})$.

Recall that our expression is

$$f(p \mid N, y) = \binom{N}{y} p^y (1-p)^{(N-y)}$$

The binomial coefficient in this equation is a constant (i.e., it does not depend on the unknown

parameter p), and so we can ignore it, and express this equation in log terms as:

$$\mathcal{L}(p \mid \text{data}) = p^y(1-p)^{(N-y)} \rightarrow \ln \mathcal{L}(p \mid \text{data}) = y \ln(p) + (N-y)\ln(1-p)$$

Note that we've written the left-hand side in a sort of short-hand notation - 'the likelihood $\mathcal{L}$ of the parameter p, given the data' (which in this case consist of 5 survivors out of 20 individuals). So, now the equation we're interested in is:

$$\ln \mathcal{L}(p \mid \text{data}) = y \ln(p) + (N-y)\ln(1-p)$$

So, all you need to do is differentiate this equation with respect to the unknown parameter p, set equal to zero, and solve.

$$\frac{\partial[\ln \mathcal{L}(p \mid \text{data})]}{\partial p} = \frac{y}{p} - \frac{(N-y)}{(1-p)} = 0$$

So, solving for p, we get:

$$\hat{p} = \frac{y}{N}$$

Thus, the value of parameter p which maximizes the likelihood of observing $y = 5$ given $N = 20$ (i.e., $\hat{p}$, the maximum likelihood estimate for p) is the same as our intuitive estimate: simply, y/N. Now, your intuition probably told you that the 'only' way you could estimate p from these data was to simply divide the number of survivors by the total number of animals. But we're sure you're relieved to learn that $5/20 = 0.25$ is also the MLE for the parameter p.

begin sidebar

closed and non-closed MLE

In the preceding example, we considered the MLE for the binomial likelihood. In that case, we could 'use algebra' to 'solve' for the parameter of interest ($\hat{p}$). When it is possible to derive an 'analytical solution' for a parameter (or set of parameters for likelihoods where there are more than one parameter), then we refer to the solution as a solution in '*closed form*'. Put another way, there is a closed form solution for the MLE for the binomial likelihood.

However, not all likelihoods have closed form solutions. Meaning, the MLE cannot be derived 'analytically' (generally, by taking the derivative of the likelihood and solving at the maximum, as we did in the binomial example). MLE's that cannot be expressed in closed form need to be solved numerically. Here is a simple example of a likelihood that cannot be put in closed form. Suppose we are interested in estimating the abundance of some population. We might intuitively understand that unless we are sure that we are encountering the entire population in our sample, then the number we encounter (the 'count' statistic; i.e., the number of individuals in our sample) is a fraction of the total population. If p is the probability of encountering any one individual in a population, and if n is the number we encounter (i.e., the number of individuals in our sample from the larger population), then we might intuitively understand that our canonical estimator for the size of the larger population is simply (n/p). For example, if there is a 50% chance of encountering an individual in a sample, and we encounter 25 individuals, then our estimate of the population size is $\hat{N} = (25/0.5) = 50$. (*Note*: we cover abundance estimation in detail in Chapter 14.)

Now, suppose you are faced with the following situation. You are sampling from a population for which you'd like to derive an estimate of abundance. We assume the population is 'closed' (no entries or exits while the population is being sampled). You go out on a number of sampling 'occasions', and capture a sample of individuals in the population. You uniquely mark each individual, and release it back into the population. At the end of the sampling, you record the total number of individuals

encountered at least once – call this M_{t+1}. Now, if the canonical estimator for abundance is $\hat{N} = (n/p)$, then $\hat{p} = (n/N)$. In other words, if we knew the size of the population N then we could derive a simple estimate of the encounter probability p by dividing the number encountered in the sample n into the size of the population. Remember, p is the probability of encountering an individual. Thus, the probability of 'missing' an individual (i.e., not encountering it) is simple $(1 - p) = 1 - (n/N)$.

So, over t samples, we can write

$$\left(1 - \frac{n_1}{N}\right)\left(1 - \frac{n_2}{N}\right) \ldots \left(1 - \frac{n_t}{N}\right) = (1 - p_1)(1 - p_2) \ldots (1 - p_t)$$

where p_i is the encounter probability at time i, and n_i is the number of individuals caught at time i.

If you think about it for a moment, you'll see that the product on right-hand side is the overall probability that an individual is not caught – not even once – over the course of the study (i.e., over t total samples). Remember from above that we defined M_{t+1} as the number of individuals caught at least once. So, we can write

$$\left(1 - \frac{M_{t+1}}{N}\right) = \left(1 - \frac{n_1}{N}\right)\left(1 - \frac{n_2}{N}\right)\left(1 - \frac{n_3}{N}\right) \cdots \left(1 - \frac{n_t}{N}\right)$$

In other words, the LHS and RHS both equal the probability of never being caught – not even once. Now, if you had estimates of p_i for each sampling occasion i, then you could write

$$\begin{aligned}
\left(1 - \frac{M_{t+1}}{N}\right) &= (1 - p_1)(1 - p_2) \ldots (1 - p_t) \\
\frac{M_{t+1}}{N} &= 1 - (1 - p_1)(1 - p_2) \ldots (1 - p_t) \\
\hat{N} &= \frac{M_{t+1}}{1 - (1 - p_1)(1 - p_2) \ldots (1 - p_t)}
\end{aligned}$$

So, the expression is rewritten in terms of N – analytical solution – closed form, right? Not quite. Note that we said *if* you had estimates of p_i. In fact, you don't. All you have is the count statistic (i.e., the number of individuals captured on each sampling occasion, n_i). So, in fact, 'all we have' are the count data (i.e., $M_{t+1}, n_1, n_2 \ldots n_t$), which (from above) we relate algebraically in the following:

$$\left(1 - \frac{M_{t+1}}{N}\right) = \left(1 - \frac{n_1}{N}\right)\left(1 - \frac{n_2}{N}\right)\left(1 - \frac{n_3}{N}\right) \cdots \left(1 - \frac{n_t}{N}\right)$$

It is not possible to 'solve' this equation so that only the parameter N appears on the LHS, while all the other terms (representing data – i.e., $M_{t+1}, n_1, n_2 \ldots n_t$) appear on the RHS. Thus, the estimator for N cannot be expressed in closed form.

However, the expression does have a solution – but it is a solution we must derive *numerically*, rather than *analytically*. In other words, we must use numerical, iterative methods to find the value of N that 'solves' this equation. That value of N is the MLE, and would be denoted as $\hat{N}$.

Consider the following data:

$$n_1 = 30, n_2 = 15, n_3 = 22, n_4 = n_t = 45, \text{ and } M_{t+1} = 79$$

Thus, one wants the value of N that 'solves' the equation

$$\left(1 - \frac{79}{N}\right) = \left(1 - \frac{30}{N}\right)\left(1 - \frac{15}{N}\right)\left(1 - \frac{22}{N}\right)\left(1 - \frac{45}{N}\right)$$

One could try to solve this equation by 'trial and error'. That is, one could plug in a guess for population size and see if the LHS = RHS (not very likely unless you can guess very well). Thinking about the problem a bit, one realizes that, logically, $N \geq M_{t+1}$ (i.e., the size of the population N must be at least as large as the number of unique individuals caught at least once, $M + t + 1$). So, at least, one

has a lower bound (in this case, 79 if we restrict the parameter space to integers). If the first guess for N does not satisfy the equation, one could try another guess and see if that either (1) satisfies the equation or (2) is closer than the first guess. The log-likelihood functions for many (but not all) problems are unimodal (for the exponential family); thus, you can usually make a new guess in the right direction.

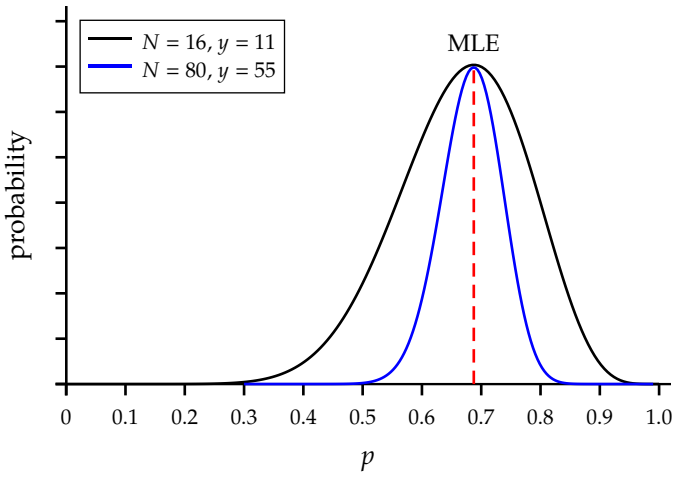
One could keep making guesses until a value of N (an integer) allows the LHS = RHS, and take this value as the MLE, $\hat{N}$. Clearly, the 'trial-and-error' method will unravel if there is more than 1 or 2 parameters. Likewise, plotting the log-likelihood function is useful only when 1 or 2 parameters are involved. We will quickly be dealing with cases where there are 30-40 parameters, thus we must rely on efficient computer routines for finding the maximum point in the multidimensional cases. Clever search algorithms have been devised for the 1-dimensional case. Computers are great at such routine computations and the MLE in this case can be found very quickly. Many (if not most) of the estimators we will work with cannot be put in closed form, and we will rely on computer software – namely, program **MARK** – to compute MLEs numerically.

end sidebar

Why go to all this trouble to derive an estimate for p? Well the maximum likelihood approach also has other uses - specifically, the ability to *estimate* the sampling variance. For example, suppose you have some data from which you have estimated that $\hat{p} = 0.6875$. Is this 'significantly different' (by some criterion) from, say, 0.5? Of course, to address this question, you need to consider the sampling variance of the estimate, since this is a measure of the uncertainty we have about our estimate. How would you do this? Of course, you might try the 'brute force' approach and simply repeat your 'experiment' a large number of times. Each time, derive the estimate of p, and then calculate a mean and variance of the parameter. While this works, there is a more elegant approach - again using ML theory and a bit more calculus (fairly straightforward stuff).

Conceptually, the sampling variance is related directly to the curvature of the likelihood at its maximum. Why? Consider the following: let's say we release 16 animals, and observe 11 survivors. What would the MLE estimate of p be? Well, we now know it is $(y/N) = (11/16) = 0.6875$. What if we had released 80 animals, instead of 16? Suppose we did this experiment, and observed 55 survivors (i.e., the expected values assuming $p = 0.6875$). What would the likelihood look like in this case? Well, clearly, the maximum of the likelihood in both 'experiments' should occur at precisely the same point: 0.6875.

But what about the 'shape' of the curve. In the following, we plot the likelihoods for both experiments ($N = 16$ and $N = 80$ respectively).



The graph shows clearly that the larger sample size ($N = 80$) results in a 'narrower' function around

the ML parameter estimate $\hat{p} = 0.6875$. If the sampling variance is related to the degree of curvature of the likelihood at its maximum, then we would anticipate the sampling variance of the parameter in these 2 experiments to be quite different, given the apparent differences in the likelihood functions.

What is the basis for stating that 'variance is related to curvature'? Think of it this way – values of the likelihood at increasing distances from the MLE are increasingly 'unlikely', relative to the MLE. The degree to which they are less likely is a function of how rapidly the curve drops away from the maximum as you move away from the MLE (i.e., the 'steepness' of the curve on either side of the MLE).

How do we address this question of 'curvature' analytically? Well, again we can use calculus. We use the first derivative of the likelihood function to find the point on the curve where the rate of change was 0 (i.e., the maximum point on the function). This first derivative of the likelihood is known as Fisher's *score function*.

We can then use the derivative of the score function with respect to the parameter(s) (i.e., the second derivative of the likelihood function, which is known as the *Hessian*), evaluated at the estimated value of the parameter (p, in this case), to 'tell us something about the curvature' at this point. In fact, more than just the curvature, Fisher showed that the negative inverse of the second partial derivative of the log-likelihood function (i.e., the negative inverse of the Hessian), evaluated at the MLE, is the MLE of the variance of the parameter. This negative inverse of the Hessian, evaluated at the MLE, is known as the *information function, or matrix*. For our example, our estimate of the variance of p is

$$\widehat{\mathrm{var}}(\hat{p}) = \left[-\left(\frac{\partial^2 \ln \mathcal{L}(p \mid \mathrm{data})}{\partial p^2} \right) \right]^{-1}_{p=\hat{p}}$$

So, we first find the second derivative of the log-likelihood (i.e., the Hessian):

$$\frac{\partial^2 \mathcal{L}}{\partial p^2} = -\frac{y}{p^2} - \frac{N-y}{(1-p)^2}$$

We evaluate this second derivative at the MLE, by substituting $y = pN$ (since $\hat{p} = y/N$). This gives

$$\left.\frac{\partial^2 \mathcal{L}}{\partial p^2}\right|_{y=pN} = -\frac{Np}{p^2} - \frac{N(1-p)}{(1-p)^2}$$

$$= -\frac{N}{p(1-p)}$$

The variance of p is then estimated as the negative inverse of this expression (i.e., the information function, or matrix), such that:

$$\widehat{\mathrm{var}}(\hat{p}) = \frac{p(1-p)}{N}$$

Some of you may recognize this as the often-used estimator of the variance of a binomial proportion - it is in all the 'stats books'. But now you can sleep more easily knowing how it was derived!

So, how do the sampling variances of our 2 experiments compare? Clearly, since p and (*1-p*) are the same in both cases (i.e., same ML estimate for $\hat{p}$), the only difference is in the denominator, N. Since $N = 80$ is obviously larger than $N = 16$, we know immediately that the sampling variance of the larger sample will be smaller (0.0027) than the sampling variance of the smaller sample (0.0134).

1.3.3. Multinomials: a simple extension

A binomial probability involves 2 possible states (e.g., live or dead). What if there are more than 2 states? In this case, we use multinomial probabilities. As with our discussion of the binomial probability (above), we start by looking at the multinomial coefficient – the multinomial equivalent of the binomial coefficient. The multinomial is extremely useful in understanding the models we'll discuss in this book. The multinomial coefficient is nearly always introduced by way of a die tossing example. So, we'll stick with tradition and discuss this classic example here. You'll recall that a die has 6 sides - therefore 6 possible outcomes if your roll a die once. The multinomial coefficient corresponding to the 'die' example is

$$\binom{N}{n_1\ n_2\ n_3\ n_4\ n_5\ n_6} = \frac{N!}{n_1!n_2!n_3!n_4!n_5!n_6!} = \frac{N!}{\prod_{i=1}^{k} n_i!}$$

Note the use of the product operator '$\prod$' in the denominator. In a multinomial context, we assume that individual trials are independent, and that outcomes are mutually exclusive and all inclusive. Consider the 'classic' die example. Assume we throw the die 60 times ($N = 60$), and a record is kept of the number of times a 1, 2, 3, 4, 5 or 6 is observed. The outcomes of these 60 independent trials are shown below.

face	*frequency*	*notation*
1	13	y_1
2	10	y_2
3	8	y_3
4	10	y_4
5	12	y_5
6	7	y_6

Each trial has a mutually exclusive outcome (1 or 2 or 3 or 4 or 5 or 6). Note that there is a type of dependency in the cell counts in that once n and y_1, y_2, y_3, y_4 and y_5 are known, then y_6 can be obtained by subtraction, because the total (N) is known. Of course, the dependency applies to any count, not just y_6. This same dependency is also seen in the binomial case - if you know the total number of coin tosses, and the total number of heads observed, then you know the number of tails, by subtraction.

The multinomial distribution is useful in a large number of applications in ecology. Its probability function for $k = 6$ is

$$P(y_i \mid n, p_i) = \binom{n}{y_i} p_1^{y_1} p_2^{y_2} p_3^{y_3} p_4^{y_4} p_5^{y_5} p_6^{y_6}$$

Again, as was the case with the binomial probability, the multinomial coefficient does not involve any of the unknown parameters, and is conveniently ignored for many estimation issues.

This is a good thing, since in the simple die tossing example the multinomial coefficient is

$$\binom{n}{y_i} = \frac{60!}{13!10!8!10!12!7!}$$

which is an absurdly big number - beyond the capacity of your simple hand calculator to calculate. So, it is helpful that we can ignore it for all intents and purposes.

Some simple examples - suppose you role a 'fair' die 6 times (i.e., 6 trials), First, assume (y_1, y_2, y_3, y_4, y_5, y_6) is a multinomial random variable with parameters $p_1 = p_2 = \ldots p_6 = 0.1667$ and $N = 6$ What

is the probability that each face is seen exactly once? This is written simply as:

$$P(1,1,1,1,1,1 \mid 6, 1/6, 1/6, 1/6, 1/6, 1/6, 1/6) = \frac{6!}{1!1!1!1!1!1!}\left(\frac{1}{6}\right)^6 = \frac{5}{324} = 0.0154$$

What is the probability that exactly four 1's occur, and two 2's occur in 6 tosses? In this case,

$$\mathcal{L}(4,2,0,0,0,0 \mid 6, 1/6, 1/6, 1/6, 1/6, 1/6, 1/6) = \frac{6!}{4!2!0!0!0!0!}\left(\frac{1}{6}\right)^4\left(\frac{1}{6}\right)^2 = \frac{5}{15{,}552} \ll 0.0154$$

As noted in our discussion of the binomial probability theorem, we are generally faced with the reverse problem – we do not know the parameters, but rather we want to estimate the parameters from the data. As we saw, these issues are the domain of the likelihood and log-likelihood functions. The key to this estimation issue is the multinomial distribution, and, particularly, the likelihood and log-likelihood functions

$$\mathcal{L}(q \mid \text{data}) \quad \text{or} \quad \mathcal{L}(p_i \mid n_i, y_i)$$

which we read as 'the likelihood of the parameters, given the data' – the left-hand expression is the more general one, where the symbol q indicates one or more parameters. The right-hand expression specifies the parameters of interest.

At first, the likelihood function looks pretty messy, but it is only a slightly different view of the probability function. Just as we saw from the binomial probability function, the multinomial function assumes N is given. The probability function further assumes that the parameters are given, while the likelihood function assumes the data are given. The likelihood function for the multinomial distribution is

$$\mathcal{L}(p_i \mid n_i, y_i) = \binom{N}{y_i} p_1^{y_1} p_2^{y_2} p_3^{y_3} p_4^{y_4} p_5^{y_5} p_6^{y_6}$$

Since the first term - the multinomial coefficient - is a constant, and since it doesn't involve any parameters, we ignore it. Next, because probabilities must sum to 1 (i.e., {sum of p_i over all i} = 1), there are only 5 'free' parameters, since the 6th one is defined by the other 5 (the 'dependency' issue we mentioned earlier), and the total, N. We will use the symbol K to denote the total number of estimable parameters in a model. Here, $K = 5$.

The likelihood function for $K = 5$, for example, is

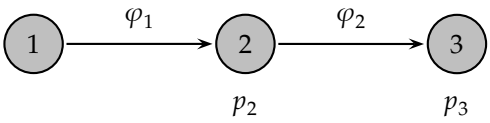
$$\mathcal{L}(p_i \mid N, y_i) = \binom{N}{y_i} p_1^{y_1} p_2^{y_2} p_3^{y_3} p_4^{y_4} p_5^{y_5} \left(1 - \sum_{i=1}^{5} p_i\right)^{\left(N - \sum_{i=1}^{5} p_i\right)}$$

So, just as we saw for the binomial example, we use a maximization routine (either analytical or numerical, depending on whether or not the likelihood can be expressed in closed form) to find the values of p_1, p_2, p_3, p_4 and p_5 that maximize the likelihood of the data that we observe. Remember - all we are doing is finding the values of the parameters which maximize the likelihood of observing the data that we see. Nothing more than that – at least conceptually.

1.4. Application to mark-recapture

Let's look at an example relevant to the task at hand (no more dice, or flipping coins.). Let's pretend we do a three year mark-recapture study, with 55 total marked individuals from a single *cohort*.* Once each year, we go out and look to see if we can 'see' (encounter) any of the 55 individuals we marked alive and in our sample. For now, we'll assume that we only encounter 'live' individuals.

The following represents the basic 'structure' of our sampling protocol.



In this diagram, each of the sampling events (referred to as 'sampling occasions') is indicated by a shaded grey circle. Our 'experiment' has three sampling occasions, numbered $1 \rightarrow 3$, respectively. In this diagram, time is moving forward going from left to right (i.e., sampling occasion 2 occurs one time step after sampling occasion 1, and so forth). Connecting the sampling occasions we have an arrow – the direction of the arrow indicates the direction of time - again, moving left to right, forward in time. We've also added two variables (symbols) to the diagram: φ and p. What do these represent?

For this example, these represent the two primary parameters which we believe (assume) govern the encounter process: φ_i (the probability of surviving from occasion i to $i+1$), and p_i (the probability that if alive and in the sample at time i, that the individual will be encountered). So, as shown on the diagram, φ_1 is the probability that an animal encountered and released alive at sampling occasion 1 will survive the interval from occasion $1 \rightarrow$ occasion 2, and so on. Similarly, p_2 is the probability that conditional on the individual being alive and in the sample, that it will be encountered at occasion 2, and so on. Why no p_1? Simple – p_1 is the probability of encountering a marked individual in the population, and none are marked prior to occasion 1 (which is when we start our study). In addition, the probability of encountering any individual (marked or otherwise) could only be calculated if we knew the size of the population, which we don't (this becomes an important consideration we will address in later chapters where we make use of estimated abundance). The important thing to remember here is the probability of being encountered at a particular sampling occasion is governed by two parameters: φ and p.

Now, as discussed earlier, if we encounter the animal, we record it in our data as '1'. If we don't encounter the animal, it's a '0'. So, based on a 3 year study, an animal with an encounter history of '111' was 'seen in the first year (the marking year), seen again in the second year, and also seen in the third year'. Compare this with an animal with an encounter history of '101'. This animal was 'seen in the first year, when it was marked, not seen in the second year, but seen again in the third year'. For a 3 occasion study, where the occasion refers to the sampling occasion, with a single release cohort, there are 4 possible encounter histories:

encounter history
111
101
110
100

* In statistics and demography, a *cohort* is a group of 'subjects' defined by experiencing a common event (typically birth) over a particular time span. In the present context, a cohort represents a group of individuals captured, marked, and released alive at the same point in time. These individuals would be part of the same *release cohort*.

Now, the key question we have to address, and (in simplest terms) the basis for analysis of data from marked individuals, is 'what is the probability of observing a particular encounter history?'. The probability of a particular encounter history is determined by a set of parameters - for this study, we know (or assume) that the parameters governing the probability of a given encounter history are φ and p. Based on the diagram on the previous page, we can write a probability expression corresponding to each of these possible encounter histories:

encounter history	*probability*
`111`	$\varphi_1 p_2 \varphi_2 p_3$
`110`	$\varphi_1 p_2 (1 - \varphi_2 p_3)$
`101`	$\varphi_1 \, (1 - p_2) \, \varphi_2 p_3$
`100`	$1 - \varphi_1 p_2 - \varphi_1 (1 - p_2) \varphi_2 p_3$

For example, take encounter history '`101`'. The individual is marked and released on occasion 1 (the first 1 in the history), is not encountered on the second occasion, but is encountered on the third occasion. Now, because of this encounter on the third occasion, we know that the individual was in fact alive on the second occasion, but simply not encountered. So, we know the individual survived from occasion $1 \rightarrow 2$ (with probability φ_1), was not encountered at occasion 2 (with probability $1 - p_2$), and survived to occasion 3 (with probability φ_2) where it was encountered (with probability p_3). So, the probability of observing encounter history '`101`' would be $\varphi_1 (1 - p_2) \varphi_2 p_3$.

Here are our 'data' - which consist of the observed frequencies of the 55 marked individuals with each of the 4 possible encounter histories:

encounter history	*frequency*
`111`	7
`110`	13
`101`	6
`100`	29

So, of the 55 individual marked and released alive in the release cohort, 7 were encountered on both sampling occasion 2 and sampling occasion 3, 13 were encountered on sampling occasion 2, but were not seen on sampling occasion 3, and so on.

The estimation problem, then, is to derive estimates of the parameters p_i and φ_i which maximizes the likelihood of observing the frequency of individuals with each of these 4 different encounter histories. Remember, the encounter histories are the data - we want to use the data to estimate the parameter values. What parameters? Again, recall also that the probability of a given encounter history is governed (in this case) by two parameters: φ, and p.

OK, so we've been playing with multinomials (above), and you might have suspected that these encounter data must be related to multinomial probabilities, and likelihoods. Good guess! The basic idea is to realize that the statistical likelihood of an actual encounter data set (as is tabulated above) is merely the product of the probabilities of the possible capture histories over those actually observed. As noted by Lebreton *et al*. (1992), because animals with the same encounter history have the same probability expression, then the number of individuals observed with each encounter history appears

as an exponent of the corresponding probability in the likelihood. Thus, we write

$$\mathcal{L} = (\varphi_1 p_2 \varphi_2 p_3)^{N_{(111)}} \times [\varphi_1 p_2 (1 - \varphi_2 p_3)]^{N_{(110)}} \times [\varphi_1 (1 - p_2) \varphi_2 p_3]^{N_{(101)}} \times [1 - \varphi_1 p_2 - \varphi_1 (1 - p_2) \varphi_2 p_3]^{N_{(100)}}$$

where $N_{(ijk)}$ is the observed frequency of individuals with encounter history ijk.

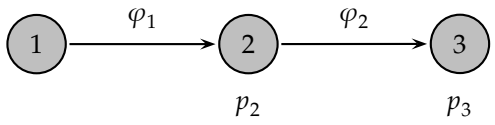
As with the binomial, we take the log transform of the likelihood expression, and after substituting the frequencies of each history, we get:

$$\ln \mathcal{L}(\varphi_1, p_2, \varphi_2, p_3) = 7 \ln(\varphi_1 p_2 \varphi_2 p_3) + 13 \ln[\varphi_1 p_2 (1 - \varphi_2 p_3)] + 6 \ln[\varphi_1 (1 - p_2) \varphi_2 p_3] + 29 \ln[1 - \varphi_1 p_2 - \varphi_1 (1 - p_2) \varphi_2 p_3]$$

All that remains is to derive the estimates of the parameters φ_i and p_i that maximize this likelihood.

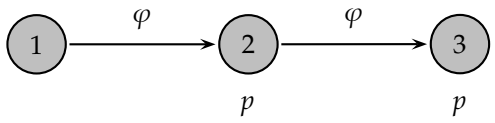
Let's go through a worked example, using the encounter history data tabulated on the preceding page. To this point, we have assumed that these encounter histories are governed by 'time-specific' variation in φ and p. In other words, we would write the probability statement for encounter history '111' as $\varphi_1 p_2 \varphi_2 p_3$.

These time-specific parameters are indicated in the following diagram:



Again, the subscripting indicates a different survival and recapture probability for each interval or sampling occasion.

However, what if instead we assume that the survival and recapture probabilities do not vary over time? In other words, $\varphi_1 = \varphi_2 = \varphi$, and $p_2 = p_3 = p$. In this case, our diagram would now look like



What would the probability statements be for the respective encounter histories?

In fact, in this case deriving them is very straightforward – we simply drop the subscripts from the parameters in the probability expressions:

encounter history	*probability*
111	$\varphi p \varphi p$
110	$\varphi p (1 - \varphi p)$
101	$\varphi (1 - p) \varphi p$
100	$1 - \varphi p - \varphi (1 - p) \varphi p$

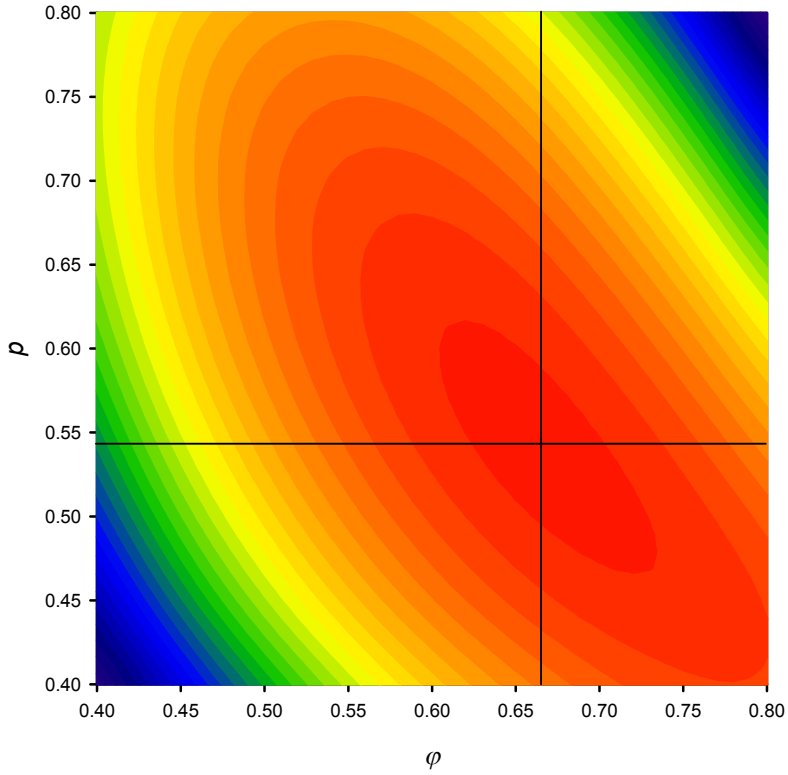
So, what would the likelihood look like? Well, given the frequencies, the likelihood would be:

$$\mathcal{L} = (\varphi p \varphi p)^{N^{111}} [\varphi p (1 - \varphi p)]^{N^{110}} [\varphi (1 - p) \varphi p]^{N^{101}} [1 - \varphi p - \varphi (1 - p) \varphi]^{N^{100}}$$

Thus,

$$\ln \mathcal{L}(\varphi, p) = 7 \ln(\varphi p \varphi p) + 13 \ln[\varphi p (1 - \varphi p)] + 6 \ln[\varphi (1 - p) \varphi p] + 29 \ln[1 - \varphi p - \varphi (1 - p) \varphi p]$$

Again, we can use numerical methods to solve for the values of φ and p which maximize the likelihood of the observed frequencies of each encounter history. The likelihood profile for these data is plotted as a 2-dimensional contour plot, shown below:



We see that the maximum of the likelihood occurs at $p = 0.542$ and $\varphi = 0.665$ (where the 2 dark black lines cross in the figure).

For this example, we used a numerical approach to find the MLE. In fact, for this example where φ and p are constant over time, the probability expressions are defined entirely by these two parameters, and we could (if we really had to) write the likelihood as two closed-form equations in φ and p, and derive estimates for φ and p analytically. All we need to do is (1) take the partial derivatives of the likelihood with respect to each of the parameters (φ, p) in turn $(\partial\mathcal{L}/\partial\varphi, \partial\mathcal{L}/\partial p)$, (2) set each partial derivative to 0, and (3) solve the resulting set of simultaneous equations.

Solving simultaneous equations is something that most symbolic math software programs (e.g., **MAPLE**, **Mathematica**, **GAUSS**, **Maxima**) does extremely well. For this problem, the MLE are estimated analytically as $\hat{\varphi} = 0.665$ and $\hat{p} = 0.542$ (just as we saw earlier using the numerical approach). However, recall that many of the likelihoods we'll be working with cannot be evaluated in closed form,

so we will rely in numerical methods. Program **MARK** evaluates all likelihoods (and functions of likelihoods) numerically.

What is the actual value of the likelihood at this point? On the log scale, $\ln(\mathcal{L})$ is maximized at -65.041. For comparison, the maximized $\ln(\mathcal{L})$ for the model where both φ and p were allowed to vary with time is -65.035. Now, these likelihoods aren't very far apart - only in the second and third decimal places. Further, the two models (with constant φ and p, and with time varying φ and p) differ by only 1 estimable parameter (we'll talk a lot more about estimable parameters in coming lectures). So, a χ^2 test would have only 1 df. The difference in the $\ln(\mathcal{L})$ is 0.006 (actually, the test is based on $2\ln(\mathcal{L})$, so the difference is actually 0.012). This difference is not significant (in the familiar sense of 'statistical significance') at $P \gg 0.5$. So, the question we now face is, which model do we pick? This takes us to one of the main themes of this book – model selection – which we'll cover in some detail in Chapter 4. But, for the moment, a glimpse of where we're headed.

1.5. Variance estimation for > 1 parameter

Earlier, we considered the derivation of the MLE and the variance for a simple situation involving only a single parameter. If in fact we have more than one parameter, the same idea we've just described for one parameter still works, but there is one important difference: a multi-parameter likelihood surface will have more than one second partial derivative. In fact, what we end up with a matrix of second partial derivatives, called the *Hessian*.

Consider for example, the log-likelihood of the simple mark-recapture data set we just analyzed in the preceding section:

$$\ln \mathcal{L}(\varphi, p) = 7\ln(\varphi p \varphi p) + 13\ln[\varphi p(1-\varphi p)] + 6\ln[\varphi(1-p)\varphi p] + 29\ln[1-\varphi p - \varphi(1-p)\varphi p]$$

Thus, the Hessian **H** (i.e., the matrix of second partial derivatives of the likelihood $\mathcal{L}$ with respect to φ and p) would be

$$\mathbf{H} = \begin{bmatrix} \dfrac{\partial^2 \mathcal{L}}{\partial \varphi^2} & \dfrac{\partial^2 \mathcal{L}}{\partial \varphi \partial p} \\ \dfrac{\partial^2 \mathcal{L}}{\partial p \partial \varphi} & \dfrac{\partial^2 \mathcal{L}}{\partial p^2} \end{bmatrix}$$

We'll leave it as an exercise for you to derive the second partial derivatives corresponding to each of the elements of the Hessian. It isn't difficult, just somewhat cumbersome.

For example,

$$\frac{\partial^2 \mathcal{L}}{\partial \varphi^2} = -\frac{26}{\varphi^2} - \frac{26p}{\varphi(1-\varphi p)} - \frac{13[p(1-\varphi p) - \varphi p^2]}{\varphi^2 p(1-\varphi p)} + \frac{13[p(1-\varphi p) - \varphi p^2]}{\varphi(1-\varphi p)^2} - \frac{58(1-p)p}{1-\varphi p - \varphi^2(1-p)p} - \frac{29[-p-2\varphi(1-p)p]^2}{[1-\varphi p - \varphi^2(1-p)p]^2}$$

Pretty ugly (and this for a simple model with only 2 parameters - φ and p - both of which are held constant over time in this example). Good thing **MARK** handles all this messy stuff for you.

Next, we evaluate the Hessian at the MLE for φ and p (i.e., we substitute the MLE values for our

parameters – $\hat{\varphi} = 0.6648$ and $\hat{p} = 0.5415$ – into the Hessian), which yields the information matrix, **I**:

$$\mathbf{I} = \begin{bmatrix} -203.06775 & -136.83886 \\ -136.83886 & -147.43934 \end{bmatrix}$$

The negative inverse of the information matrix ($-\mathbf{I}^{-1}$) is the variance-covariance matrix for parameters φ and p

$$-\mathbf{I}^{-1} = -\begin{bmatrix} -203.06775 & -136.83886 \\ -136.83886 & -147.43934 \end{bmatrix}^{-1} = \begin{bmatrix} 0.0131 & -0.0122 \\ -0.0122 & 0.0181 \end{bmatrix}$$

Note that the variances are found along the diagonal of the matrix, while the off-diagonal elements are the covariances.

In general, for an arbitrary parameter θ, the variance of θ_i is given as the elements of the negative inverse of the information matrix corresponding to

$$\frac{\partial^2 \ln \mathcal{L}}{\partial \theta_i \partial \theta_i}$$

while the covariance of θ_i with θ_j is given as the elements of the negative inverse of the information matrix corresponding to

$$\frac{\partial^2 \ln \mathcal{L}}{\partial \theta_i \partial \theta_j}$$

Obviously, the variance-covariance matrix is the basis for deriving measures of the precision of our estimates. But, as we'll see in later chapters, the variance-covariance matrix is used for much more - including estimating the number of estimable parameters in the model. While **MARK** handles all this for you, it's important to have a least a feel for what **MARK** is doing 'behind the scenes', and why.

1.6. More than 'estimation' - ML and statistical testing

In the preceding, we focussed on the maximization of the likelihood as a means of deriving estimates of parameters and the sampling variance of those parameters. However, the other primary use of likelihood methods is for comparing the fits of different models.

We know that $\mathcal{L}(\hat{\theta})$ is the value of the likelihood function evaluated at the MLE $\hat{\theta}$, whereas $\mathcal{L}(\theta)$ is the likelihood for the true (but unknown) parameter θ. Since the MLE maximizes the likelihood for a given sample, then the value of the likelihood at the true parameter value θ is generally smaller than the MLE $\hat{\theta}$ (unless by chance $\hat{\theta}$ and θ happen to coincide).

This, combined with other properties of ML estimators noted earlier lead directly to several classic and general procedures for testing the statistical hypothesis that $H_o : \theta = \theta_0$. Here we briefly describe three of the more commonly used tests.

Fisher's Score Test

The 'score' is the slope of the log-likelihood at a particular value of θ. In other words, $S(\theta) = \partial \ln \mathcal{L}(\theta)/\partial\theta$. At the MLE, the score (slope) is 0 (by definition of a maximum).

Recall from earlier in this chapter that

$$\widehat{\text{var}}(\hat{\theta}) = \left[-\left(\frac{\partial^2 \ln \mathcal{L}(\theta \mid data)}{\partial \theta^2} \right) \right]^{-1}_{\theta=\hat{\theta}}$$

The term inside the inner parentheses is known as *Fisher information*

$$I(\theta) = -\frac{\partial^2 \ln \mathcal{L}(\theta)}{\partial \theta^2}$$

It can be shown that the score statistic

$$S_0 = \frac{S(\theta_0)}{\sqrt{I(\theta_0)}}$$

is asymptotically distributed as $N(0,1)$ under H_0.

Wald test

The Wald test relies on the asymptotic normality of the MLE $\hat{\theta}$. Given the normality of the MLE, we can calculate the test statistic

$$Z_0 = \frac{\hat{\theta} - \theta_0}{\sqrt{\widehat{\text{var}}(\hat{\theta})}}$$

which is asymptotically distributed as $N(0,1)$ under the null H_o.

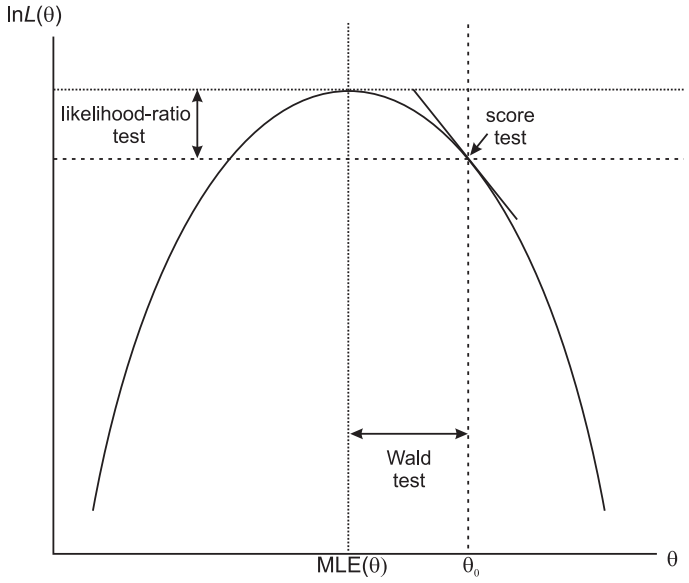
Likelihood ratio test

It is known that

$$2\left[\ln \mathcal{L}(\hat{\theta}) - \ln \mathcal{L}(\theta_0)\right]$$

follows an asymptotic χ^2 distribution with one degree of freedom.

The basic relationship among these tests is shown in the following diagram:



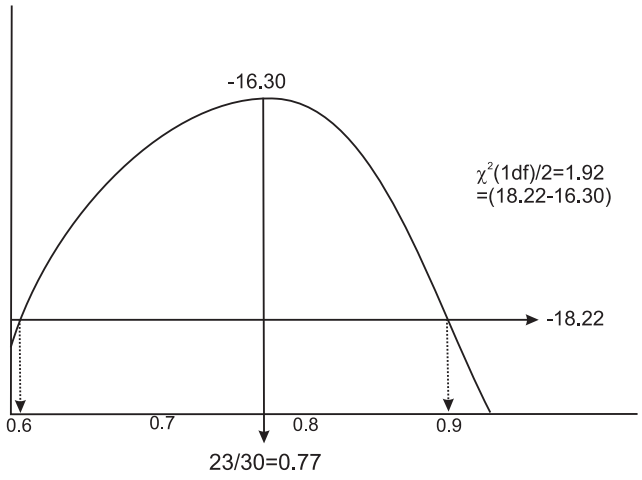
In general, these three tests are asymptotically equivalent, although in some applications, the score test has the practical advantage of not requiring the computation of the MLE at $\hat{\theta}$ (since S_0 depends only on the null value θ_0, which is specified in H_0). We consider one of these tests (the likelihood ratio test) in much more detail in Chapter 4.

1.7. Technical aside: a bit more on variances

As we discussed earlier, the classic MLE approach to variance calculation (for purposes of creating a SE and so forth) is to use the negative inverse of the 2nd derivative of the MLE evaluated at the MLE. However, the problem with this approach is that, in general, it leads to derivation of symmetrical 95% CI, and in many cases – especially for parameters that are bounded [0, 1] – this makes no sense. A simple example will show what we mean. Suppose we release 30 animals, and find 1 survivor. We know from last time that the MLE for the survival probability is (1/30) = 0.03333. We also know from earlier in this chapter that the classical estimator for the variance, based on the 2nd derivative, is

$$\widehat{\text{var}}(\hat{p}) = \frac{p(1-p)}{N} = \frac{0.0333(1-0.0333)}{30} = 0.0010741$$

So, based on this, the 95% CI using classical approaches would be ±1.96(SE), where the SE (standard error) is estimated as the square-root of the variance. Thus, given $\widehat{\text{var}} = 0.001074$, the 95% CI would be ±1.96(0.03277), or [0.098, −0.031]. OK, so what's wrong with this? Well, clearly, we don't expect a 95% CI to ever allow values < 0 (or > 1) for a parameter that is logically bounded to fall between 0 and 1 (like φ or p). So, there must be a problem, right? Well, somewhat. Fortunately, however, there is a better way, using something called the *profile likelihood* approach, which makes more explicit use of the shape of the likelihood. We'll go into the profile likelihood in further detail in later chapters, but to briefly introduce the concepts - consider the following diagram, which shows the maximum part of the log likelihood for φ, given $N = 30$, $y = 23$ (i.e., 23/30 survive).



Profile likelihood confidence intervals are based on the log-likelihood function. For a single parameter, likelihood theory shows that the 2 points 1.92 units down from the maximum of the log likelihood function provide a 95% confidence interval when there is no extra-binomial variation (i.e., $c = 1$; see Chapter 5). The value 1.92 is half of the $\chi^2_1 = 3.84$. Thus, the same confidence interval can be computed

with the *deviance* by adding 3.84 to the minimum of the deviance function, where the deviance is the log-likelihood multiplied by -2 minus the -2 log likelihood value of the saturated model (more and these concepts in later chapters).

Put another way, we use the critical value of 1.92 to derive the *profile* - you take the value of the log likelihood at the maximum (for this example, the maximum occurs at −16.30), add 1.92 to it (yielding −18.22; note we keep the negative sign here), and look to see where the −18.22 line intersects with the *profile* of the log likelihood function. In this case, we see that the intersection occurs at approximately 0.6 and 0.9. The MLE is (23/30) = 0.767, so clearly, the profile 95% CI is not symmetrical around this MLE value. But, it **is** bounded on the interval [0, 1]. The profile likelihood is the preferred approach to deriving 95% CI. The biggest limit to using it is computational – it simply takes more work to derive a profile likelihood (and corresponding CI). Fortunately, **MARK** does all the work for us.

1.8. Summary

That's it for Chapter 1! Nothing about **MARK**, but some important background. Beginning with Chapter 2, we'll consider formatting of our data (the 'encounter histories' we introduced briefly in this chapter). After that, the real details of using program **MARK**. Our suggestion at this stage is to (i) leave your own data alone – you need to master the basics first. This means working through at least chapters 3 → 8, in sequence, using the example data sets. Chapter 9 and higher refer to specific data types – one or more may be of particular interest to you. Then, when you're ready (i.e., have a good understanding of the basic concepts), (ii) get your data in shape – this is covered in Chapter 2.

CHAPTER 2

Data formatting: the input file . . .

Clearly, the first step in any analysis is gathering and collating your data. We'll assume that at the minimum, you have records for the individually marked individuals in your study, and from these records, can determine whether or not an individual was 'encountered' (in one fashion or another) on a particular sampling occasion. Typically, your data will be stored in what we refer to as a 'vertical file' - where each line in the file is a record of when a particular individual was seen. For example, consider the following table, consisting of some individually identifying mark (ring or tag number), and the year. Each line in the file (or, row in the matrix) corresponds to the animal being seen in a particular year.

tag number	*year*
1147-38951	73
1147-38951	75
1147-38951	76
1147-38951	82
1147-45453	74
1147-45453	78

However, while it is easy and efficient to record the observation histories of individually marked animals this way, the 'vertical format' is not at all useful for capture-mark-recapture analysis. The preferred format is the *encounter history*. The encounter history is a contiguous series of specific dummy variables, each of which indicates something concerning the encounter of that individual - for example, whether or not it was encountered on a particular sampling occasion, how it was encountered, where it was encountered, and so forth. The particular encounter history will reflect the underlying model type you are working with (e.g., recaptures of live individuals, recoveries of dead individuals). Consider for example, the encounter history for a typical mark-recapture analysis (the encounter history for a mark-recapture analysis is often referred to as a *capture history*, since it implies physical capture of the individual). In most cases, the encounter history consists of a contiguous series of '1's and '0's, where '1' indicates that an animal was recaptured (or otherwise known to be alive and in the sampling area), and '0' indicates the animal was not recaptured (or otherwise seen). Consider the individual in the preceding table with tag number '1147-38951'. Suppose that 1973 is the first year of the study, and that 1985 is the last year of the study. Examining the table, we see that this individual was captured and marked during the first year of the study, was seen periodically until 1982, when it was seen for the last time. The corresponding encounter-history for this individual would be: '1011000001000'.

In other words, the individual was seen in 1973 (the starting '1'), not seen in 1974 ('0'), seen in 1975 and 1976 ('11'), not seen for the next 5 years ('00000'), seen again in 1982 ('1'), and then not seen again

('000').

While this is easy enough in principal, you surely don't want to have to construct capture-histories manually. Of course, this is precisely the sort of thing that computers are good for - large-scale data manipulation and formatting. **MARK** does not do the data formatting itself - no doubt you have your own preferred 'data manipulation' environment (**dBASE**, **Excel**, **Paradox**, **SAS**). Thus, in general, you'll have to write your own program to convert the typical 'vertical' file (where each line represents the encounter information for a given individual on a given sampling occasion; see the example on the preceding page) into encounter histories (where the encounter history is a horizontal string). In fact, if you think about it a bit, you realize that in effect what you need to do is to take a vertical file, and 'transpose' (or, 'pivot') it into a horizontal file - where fields to the right of the individual tag number represent when an individual was recaptured or resighted. However, while the idea of a 'transpose' or 'pivot' seems simple enough, there is one rather important thing that needs to be done - your program must insert the '0' value whenever an individual was not seen. We'll assume for the purposes of this book that you will have some facility to put your data into the proper encounter-history format. For those of you who have no idea whatsoever on how to approach this problem, we provide some practical guidance in the Addendum at the end of this chapter. Of course, you could always do it by hand, if absolutely necessary!

begin sidebar

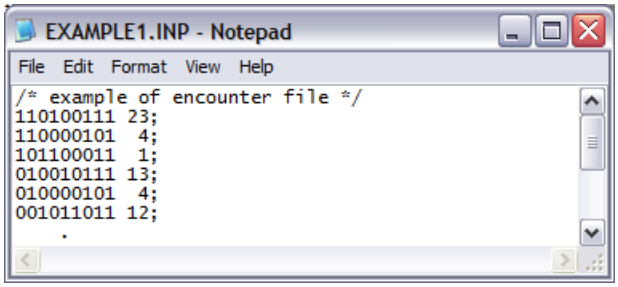
editing the .INP file

Many of the problems people have getting started with **MARK** can ultimately be traced back to problems with the .INP file. One common issue relates to choice of editor used to make changes/aditions to the .INP file. You are strongly urged to avoid - as in 'like the plague' - using Windows Notepad (or, even worse, Word) to do much of anything related to building/editing .INP files. Do yourself a favor and get yourself a real ASCII editor - there are a number of very good 'free' applications you can (and should) use instead of Notepad (e.g., Notepad++, EditPad Lite, jEdit, and so on...)

end sidebar

2.1. Encounter histories formats

Now we'll look at the formatting of the encounter histories file in detail. It is probably easiest to show you a 'typical' encounter history file, and then explain it 'piece by piece'. The encounter-history reflects a mark-recapture experiment.



Superficially, the encounter histories file is structurally quite simple. It consists of an ASCII (text) file, consisting of the encounter history itself (the contiguous string of dummy variables), followed by one or more additional columns of information pertaining to that history. Each record (i.e., each

line) in the encounter histories file ends with a semi-colon. Each history (i.e., each line, or record) must be the same length (i.e., have the same number of elements - the encounter history itself must be the same length over all records, and the number of elements 'to the right' of the encounter history must also be the same) - this is true regardless of the data type. The encounter histories file should have a `.INP` suffix (for example, `EXAMPLE1.INP`). Generally, there are no other 'control statements' or 'PROC statements' required in a **MARK** input file. However, you can optionally add comments to the `INP` file using the 'slash-asterisk asterisk/slash' convention common to many programming environments - we have included a comment at the top of the example input file (shown at the bottom of the preceding page). The only thing to remember about comments is that they do **not** end with a semi-colon.

Let's look at each record (i.e., each line) a bit more closely. In this example, each encounter history is followed by a number. This number is the frequency of all individuals having a particular encounter history. This is not required (and in fact isn't what you want to do if you're going to consider individual covariates - more on that later), but is often more convenient for large data sets. For example, the summary encounter history

```
110000101 4;
```

could also be entered in the `INP` files as

```
110000101 1;
110000101 1;
110000101 1;
110000101 1;
```

Note again that each line – each 'encounter history record' – ends in a semi-colon. How would you handle multiple groups? For example, suppose you had encounter data from males and females? In fact, it is relatively straightforward to format the `INP` file for multiple groups - very easy for summary encounter histories, a bit less so for individual encounter histories. In the case of summary encounter histories, you simply add a second column of frequencies to the encounter histories to correspond to the other sex. For example,

```
110100111 23 17;
110000101  4  2;
101100011  1  3;
```

In other words, 23 of one sex and 17 of the other have history '`110100111`' (the ordering of the sexes - which column of frequencies corresponds to which sex - is entirely up to you). If you are using individual records, rather than summary frequencies, you need to indicate group association in a slightly less-obvious way - you will have to use a '`0`' or '`1`' within a group column to indicate the frequency - but obviously for one group only. We'll demonstrate the idea here. Suppose we had the following summary history, with frequencies for males and females (respectively):

```
110000101   4   2;
```

In other words, 4 males, and 2 females with this encounter history (note: the fact that males come before females in this example is completely arbitrary. You can put whichever sex - or 'group' - you want in any column you want - all you'll need to do is remember which columns in the `INP` file correspond to which groups).

To 'code' individual encounter histories, the INP file would be modified to look like:

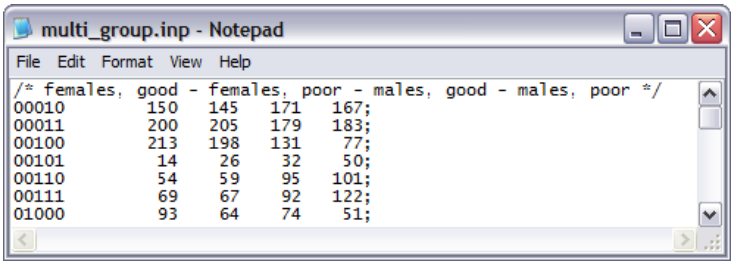
```
110000101  1 0;
110000101  1 0;
110000101  1 0;
110000101  1 0;
110000101  0 1;
110000101  0 1;
```

In this example, the coding '1 0' indicates that the individual is a male (frequency of 1 in the male column, frequency of 0 in the female column), and '0 1' indicates the individual is a female (frequency of 0 in the male column, and frequency of 1 in the male column). The use of one-record per individual is only necessary if you're planning on using individual covariates in your analysis.

2.1.1. Groups within groups...

In the preceding example, we had 2 groups: males and females. The frequency of encounters for each sex is coded by adding the frequency for each sex to the right of the encounter history.

But, what if you had something like males, and females (i.e., data from both sexes) and good colony and poor colony (i.e., data were sampled for both sexes from each of 2 different colonies - one classified as good, and the other as poor). How do you handle this in the INP file? Well, all you need to do is have a frequency column for each (sex.colony) combination: one frequency column for females from the good colony, one frequency column for females from the poor colony, one frequency column for males from the good colony, and finally, one frequency column for males from the poor colony. An example of such an INP file is shown below:



As we will see in subsequent chapters, building models to test for differences between and among groups, and for interactions among groups (e.g., an interaction of sex and colony in this example) is relatively straightforward in **MARK** - all you'll really need to do is remember which frequency column codes for which grouping (hence the utility of adding comments to your INP file, as we've done in this example).

2.2. Removing individuals from the sample

Occasionally, you may choose to remove individuals from the data set at a particular sampling occasion. For example, because your experiment requires you to remove the individual after its first recapture, or because it is injured, or for some other reason. The standard encounter history we have looked at so far records presence or absence only. How do we accommodate 'removals' in the INP file? Actually,

it's very easy - all you do is change the 'sign' on the frequencies from positive to negative. Negative frequencies indicates that that many individuals with a given encounter history were removed from the study. For example,

```
100100 1500 1678;
100100  -23  -25;
```

In this example, we have 2 groups, and 6 sampling occasions. In the first record, we see that there were 1,500 individuals and 1,678 individuals in each group marked on the first occasion, not encountered on the next 2 occasions, seen on the fourth occasion, and not seen again. In the second line, we see the same encounter history, but with the frequencies '-23' and '-25'. The negative values indicate to **MARK** that 23 and 25 individuals in both groups were marked on the first occasion, not seen on the next 2 occasions, were encountered on the fourth occasion, at which time they were removed from the study. Clearly, if they were removed, they cannot have been seen again.

begin sidebar

uneven time-intervals between sampling occasions?

In the preceding, we have implicitly assumed that the sampling interval between sampling occasions is identical throughout the course of the study (e.g., sampling every 12 months, or every month, or every week). But, in practice, it is not uncommon for the time interval between occasions to vary – either by design, or because of 'logistical constraints'. This has clear implications for how you analyze your data. For example, suppose you sample a population each October, and again each May (i.e., two samples within a year, with different time intervals between samples; October→ May (7 months), and May→ October (5 months)). Suppose the true monthly survival rate is constant over all months, and is equal to 0.9. As such, the estimated survival for October→ May will be $0.9^7 = 0.4783$, while the estimated survival rate for May→ October will be $0.9^5 = 0.5905$. Thus, if you fit a model without accounting for these differences in time intervals, it is clear that there would 'appear' to be differences in survival between successive samples, when in fact the monthly survival does not change over time.

So, how do you 'tell **MARK**' that the interval between samples may vary over time? You might think that you need to 'code' this interval information in the `.INP` file in some fashion. In fact, you don't - you specify the time intervals when you are specifying the data type in **MARK**, and not in the `.INP` file. In the `.INP` file, you simply enter the encounter histories as contiguous strings, regardless of the true interval between sampling occasions. We will discuss handling uneven time-intervals in more detail in a later chapter.

end sidebar

2.3. Different encounter history formats

Up until now, we've more or less used typical mark-recapture encounter histories (i.e., capture histories) to illustrate the basic principles of constructing an `.INP` file. However, **MARK** can be applied to far more than mark-recapture analyses, and as such, there are a number of slight permutations on the encounter history that you need to be aware of in order to use **MARK** to analyze your particular data type. First, we summarize in table form (on the next page) the different data types **MARK** can handle, and the corresponding encounter history format.

recaptures only	LLLL
recoveries only	LDLDLDLD
both	LDLDLDLD
known fate	LDLDLDLD
closed captures	LLLL
BTO ring recoveries	LDLDLDLD
robust design	LLLL
both (Barker model)	LDLDLDLD
multi-strata	LLLL
Brownie recoveries	LDLDLDLD
Jolly-Seber	LLLL
Huggins' closed captures	LLLL
Robust design (Huggins)	LLLL
Pradel recruitment	LLLL
Pradel survival & seniority	LLLL
Pradel survival & λ	LLLL
Pradel survival & recruitment	LLLL
POPAN	LLLL
multi-strata - live and dead encounters	LDLDLDLD
closed captures with heterogeneity	LLLL
full closed captures with heterogeneity	LLLL
nest survival	LDLDLDLD
occupancy estimation	LLLL
robust design occupancy estimation	LLLL
open robust design multi-strata	LLLL
closed robust design multi-strata	LLLL

Each data type in **MARK** requires a primary from of data entry provided by the encounter history. Encounter histories can consist of information on only live encounters (LLLL) or information on both live and dead (LDLDLDLD). In addition, some types allow a summary format (e.g., recovery matrix) which reduces the amount of input. The second column of the table shows the basic structure for a 4 occasion encounter history. There are, in fact, broad types: live encounters only, and mixed live and dead (or known fate) encounters. For example, for a recaptures only study (i.e., live encounters), the structure of the encounter history would be 'LLLL' - where 'L' indicates information on encountered/not encountered status. As such, each '**L**' in the history would be replaced by the corresponding 'coding variable' to indicate encountered or not encountered status (usually '1' or '0' for the recaptures only history). So, for example, the encounter '1011' indicates seen and marked alive at occasion 1, not seen on occasion 2, and seen again at both occasion 3 and occasion 4. For data types including both live and dead individuals, the encounter history for the 4 occasion study is effectively 'doubled' - taking the format 'LDLDLDLD', where the 'L' refers to the live encountered or not encountered status, and the 'D' refers to the dead encountered or not encountered status. At each sampling occasion, either 'event' is possible - an individual could be both seen alive at occasion (i) and then found dead at occasion (i), or during the interval between (i) and (i+1). Since both 'potential events' need to be coded at each occasion, this effectively doubles the length of the encounter history from a 4 character string to an 8 character string.

For example, suppose you record the following encounter history for an individual over 4 occasions - where the encounters consist of both live encounters and dead recoveries. Thus, the history '10001100' reflects an individual seen and marked alive on the first occasion, not recovered during the first interval,

not seen alive at the second occasion and not recovered during the second interval, seen alive on the third occasion and then recovered dead during the third interval, and not seen or recovered thereafter (obviously, since the individual was found dead during the preceding interval).

2.4. Some more examples

The **MARK** help files contain a number of different examples of encounter formats. We list only a few of them here. For example, suppose you are working with dead recoveries only. If you look at the table on the preceding page, you see that it has a format of 'LDLDLDLD'. Why not just 'LLLL', and using '1' for live', and '0' for recovered dead? The answer is because you need to differentiate between known dead (which is a known fate) , and simply not seen. '0' alone could ambiguously mean either dead, or not seen (or both!).

2.4.1. Dead recoveries only

The following is an example of dead recoveries only, because a live animal is never captured alive after its initial capture. That is, none of the encounter histories have more than one '1' in an **L** column. This example has 15 encounter occasions and 1 group. If you study this example, you will see that 500 animals were banded each banding occasion.

```
000000000000000000000000000010  465;
000000000000000000000000000011   35;
000000000000000000000000001000  418;
000000000000000000000000001001   15;
000000000000000000000000001100   67;
000000000000000000000000100000  395;
000000000000000000000000100001    3;
000000000000000000000000100100   25;
000000000000000000000000110000   77;
```

Traditionally, recoveries only data sets were summarized into what are known as recovery tables. **MARK** accommodates *recovery tables*, which have a 'triangular matrix form', where time goes from left to right (shown below). This format is similar to that used by Brownie *et al*. (1985).

```
 7    4   1    0   1;
      8   5    1   0;
         10    4   2;
               16   3;
                   12;
99   88 153  114 123;
```

Following each matrix is the number of individuals marked each year. So, 99 individuals marked on the first occasion, of which 7 were recovered dead during the first interval, 4 during the second, 1 during the third, and so on.

2.4.2. Individual covariates

Finally, an example of known fate data, where individual covariates are included. Comments are given at the start of each line to identify the individual (this is optional, but often very helpful in keeping track of things). Then comes the capture history for this individual, in a 'LDLDLD. . .' sequence. Thus the first capture history is for an animal that was released on occasion 1, and died during the interval. The second animal was released on occasion 1, survived the interval, released again on occasion 2, and died during this second interval. Following the capture history is the count of animals with this history (always 1 in this example). Then, 4 covariates are provided. The first is a dummy variable representing age (0=subadult, 1=adult), then a condition index, wing length, and body weight.

```
/* 01 */     1100000000000000     1    1   1.16   27.7  4.19;
/* 04 */     1011000000000000     1    0   1.16   26.4  4.39;
/* 05 */     1011000000000000     1    1   1.08   26.7  4.04;
/* 06 */     1010000000000000     1    0   1.12   26.2  4.27;
/* 07 */     1010000000000000     1    1   1.14   27.7  4.11;
/* 08 */     1010110000000000     1    1   1.20   28.3  4.24;
/* 09 */     1010000000000000     1    1   1.10   26.4  4.17;
```

What if you have multiple groups, such that individuals are assigned (or part of) a given group, and where you also have individual covariates? There are a couple of ways you could handle this sort of situation. You can either code for the groups explicitly in the .inp file, or use an individual covariate for the groups. There are pros and cons to either approach (this issue is discussed in Chapter 11).

Here is an snippet from a data set with 2 groups coded explicitly, and an individual covariate. In this data fragment, the first 8 contiguous values represent the encounter history, followed by 2 columns representing the frequencies depending on group: '1 0' indicating group 1, and '0 1' indicating group 2, followed by the value of the covariate:

```
11111111 1 0 123.211;
11111111 0 1  92.856;
11111110 1 0 122.115;
11111110 1 0 136.460;
```

So, the first record with an encounter history of '11111111' is in group 1, and has a covariate value of 123.211. The second individual, also with an encounter history of '11111111', is in group 2, and has a covariate value of 92.856. The third individual has an encounter history of '11111110', and is in group 1, with a covariate value of 122.115. And so on.

If you wanted to code the group as an individual covariate, this same input file snippet would look like:

```
11111111 1 1 123.211;
11111111 1 0  92.856;
11111110 1 1 122.115;
11111110 1 1 136.460;
```

In this case, following the encounter history, is a column of 1's, indicating the frequency for each individual, followed by a column containing a 0/1 dummy code to indicate group (in this example, we've used a 1 to indicate group 1, 0 to indicate group 2), followed by the value of the covariate.

A final example – for three groups where we code for each group explicitly (such that each group has it's own 'dummy column' in the input file), an encounter history with individual covariates might look like:

```
11111  1 0 0  123.5;
11110  0 1 0   99.8;
11111  0 0 1  115.2;
```

where the first individual with encounter history '11111' is in group 1 (dummy value of 1 in the first column after the encounter history, and 0's in the next two columns) and has a covariate value of 123.5, second individual with encounter history '11110' is in group 2 (dummy code of 0 in the first column, 1 in the second, and 0 in the third) and a covariate value of 99.8, and a third individual with encounter history '11111' in group 3 (0 in the first two columns, and a 1 in the third column), with a covariate value of 115.2.

As is noted in the help file (and discussed at length in Chapter 11), it is helpful to scale the values of covariates to have a mean on the interval [0, 1] to ensure that the numerical optimization algorithm finds the correct parameter estimates. For example, suppose the individual covariate '`weight`' is used, with a range from 1,000 g to 5,000 g. In this case, you should scale the values of weight to be from 0.1 to 0.5 by multiplying each 'weight' value by 0.0001. In fact, **MARK** defaults to doing this sort of scaling for you automatically (without you even being aware of it). This 'automatic scaling' is done by determining the maximum absolute value of the covariates, and then dividing each covariate by this value. This results in each column scaled to between -1 and 1. This internal scaling is purely for purposes of ensuring the success of the numerical optimization - the parameter values reported by **MARK** (i.e., in the output that you see) are 'back-trasformed' to the original scale. Alternatively, if you prefer that the 'scaled' covariates have a mean of 0, and unit variance (this has some advantages in some cases), you can use the '**`Standardize Individual Covariates`**' option of the '**`Run Window`**' to perform the default standardization method (more on these in subsequent chapters).

More details on how to handle individual covariates in the input file are given in Chapter 11.

Summary

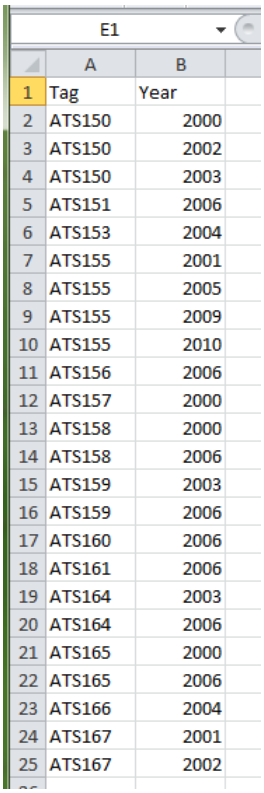
That's it! You're now ready to learn how to use **MARK**. Before you leap into the first major chapter (Chapter 3), take some time to consider that **MARK** will always do its 'best' to analyze the data you feed into it. However, it assumes that you will have taken the time to make sure your data are correct. If not, you'll be the unwitting victim to perhaps the most telling comment in data analysis: 'garbage in...garbage out'. Take some time at this stage to make sure you are confident in how to properly create and format your files.

Addendum: generating .inp files

Andrew Sterner, *Marine Turtle Research Group, University of Central Florida*

As noted at the outset in this chapter, **MARK** has no capability of generating input (`.INP`) files. This is something you will need to do for yourself. In this short addendum, we introduce one approach to generating `.INP` files, based on 'Excel pivot tables'. Since there are any number of different software applications for managing and manipulating data, we state for the record that we are going to demonstrate creating `.INP` files using Excel, not as a point of advocacy for using Excel, but owing more to its near ubiquity (*note*: most of what follows applies generally to Access databases as well).

We will demonstrate the basic idea using an example where we will reformat an Excel spreadsheet containing some live encounter data. We wish to format these data into an `.INP` file. The data are contained in the Excel spreadsheet `csj-pivot.xlsx` (note, we're clearly using Excel 2007 or later). Here are what the data look like before we transform them into an input file.



	A	B
1	Tag	Year
2	ATS150	2000
3	ATS150	2002
4	ATS150	2003
5	ATS151	2006
6	ATS153	2004
7	ATS155	2001
8	ATS155	2005
9	ATS155	2009
10	ATS155	2010
11	ATS156	2006
12	ATS157	2000
13	ATS158	2000
14	ATS158	2006
15	ATS159	2003
16	ATS159	2006
17	ATS160	2006
18	ATS161	2006
19	ATS164	2003
20	ATS164	2006
21	ATS165	2000
22	ATS165	2006
23	ATS166	2004
24	ATS167	2001
25	ATS167	2002

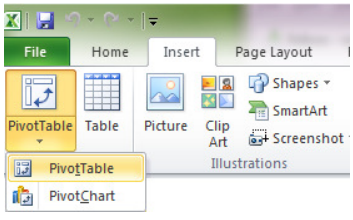
The file consists of two data columns: `TAG` (indicating the individual), and `YEAR` (the year that the individual was encountered). This data file contains encounter data for 14 marked individuals, with encounter data collected from 2000 to 2010 (thus, the encounter history will be 11 characters in length).

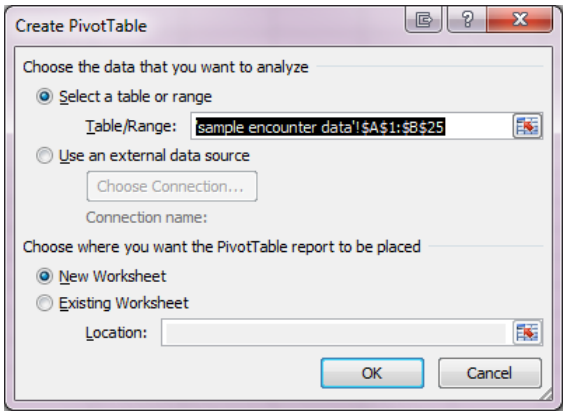
Our challenge, then is to take this 'vertical' file (one record per individual each year encountered), and 'pivot' it horizontally. For example, take the first individual in the file, `ATS150`. It was first encountered in 2000, again in 2002, and again (for the final time) in 2003. The second individual, `ATS151`, was seen for the first time in 2006, and then not seen again. The third individual, `ATS153`, was seen in 2004, and not seen again after that. And so on. If we had to generate the `.INP` file by hand for these individuals, their encounter histories would look like:

```
/* ATS150 */ 101100000   1;
/* ATS151 */ 000000100   1;
/* ATS153 */ 000010000   1;
```

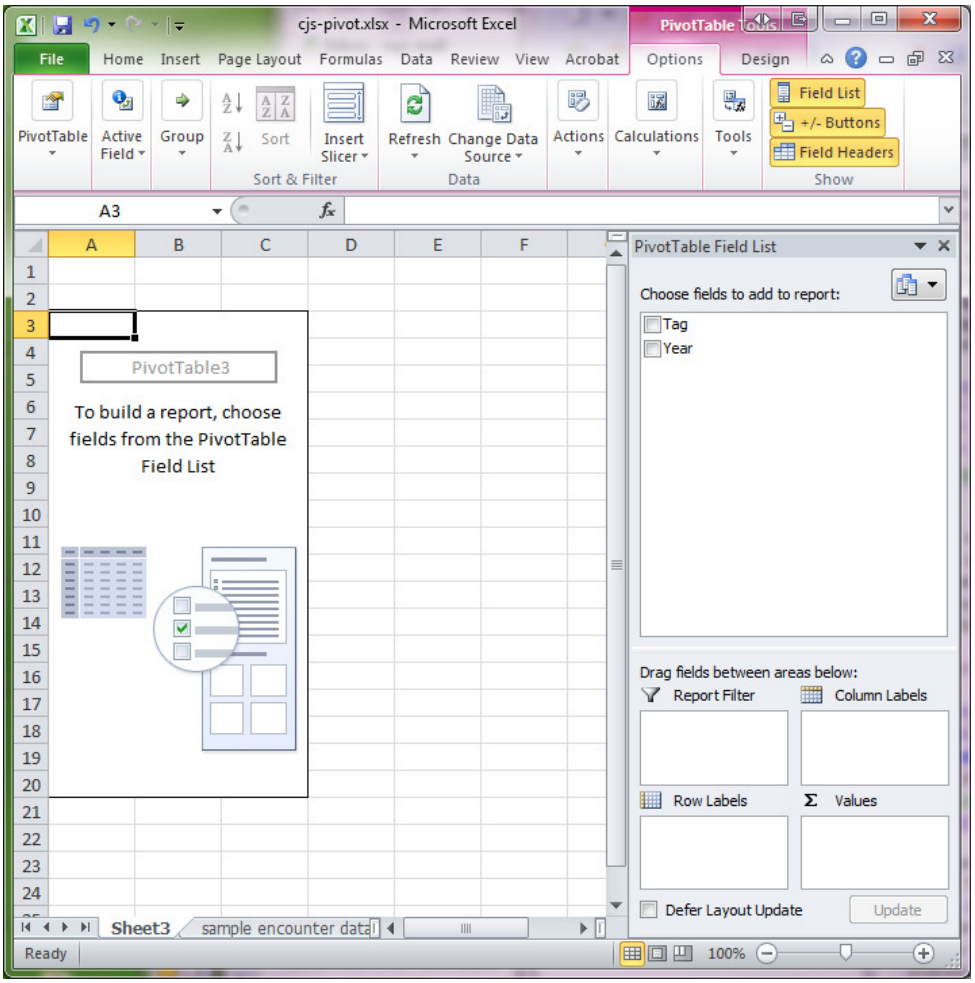
As it turns out, we can make use of the 'pivot table' in Excel (and some simple steps involving 'search and replace' and the `CONCATENATE` function), to generate exactly what we need. The process can be more involved for more complicated data types (e.g., robust design), but the basic principle of 'pivoting' applies.

Here are the basic steps. First, we select the rows and columns containing the data. Then, select '**`Insert | PivotTable | PivotTable`**', as shown to the right (make sure you select **`PivotTable`** and not **`PivotChart`**). This will bring up a dialog window (shown at the top of the next page) asking you to choose the data you want to 'pivot', and where you want the pivot table to be placed.

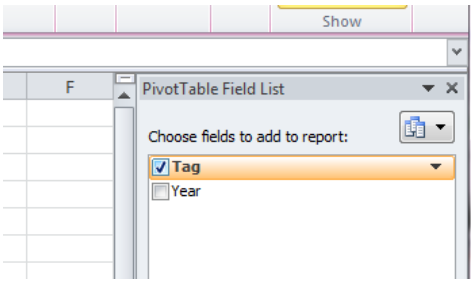




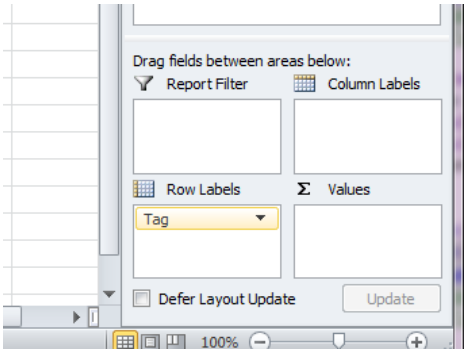
The '**`Table/Range`**' field will already be filled with the rows and columns of the data you selected. We strongly recommend you put the pivot table into a '**`New Worksheet`**' (this is selected by default). Once you click '**`OK`**', you will be presented with the template from which you will generate the pivot table:



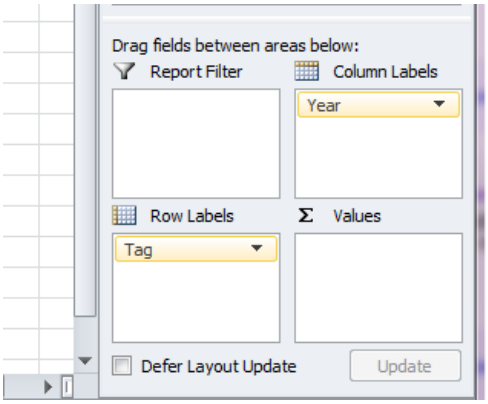
All you really need to do at this point is specify the '**`row labels`**', the '**`column labels`**', and the '**`values`**' (on the right hand side of the template). So, to specify the row labels, we simply select '**`tag`**'



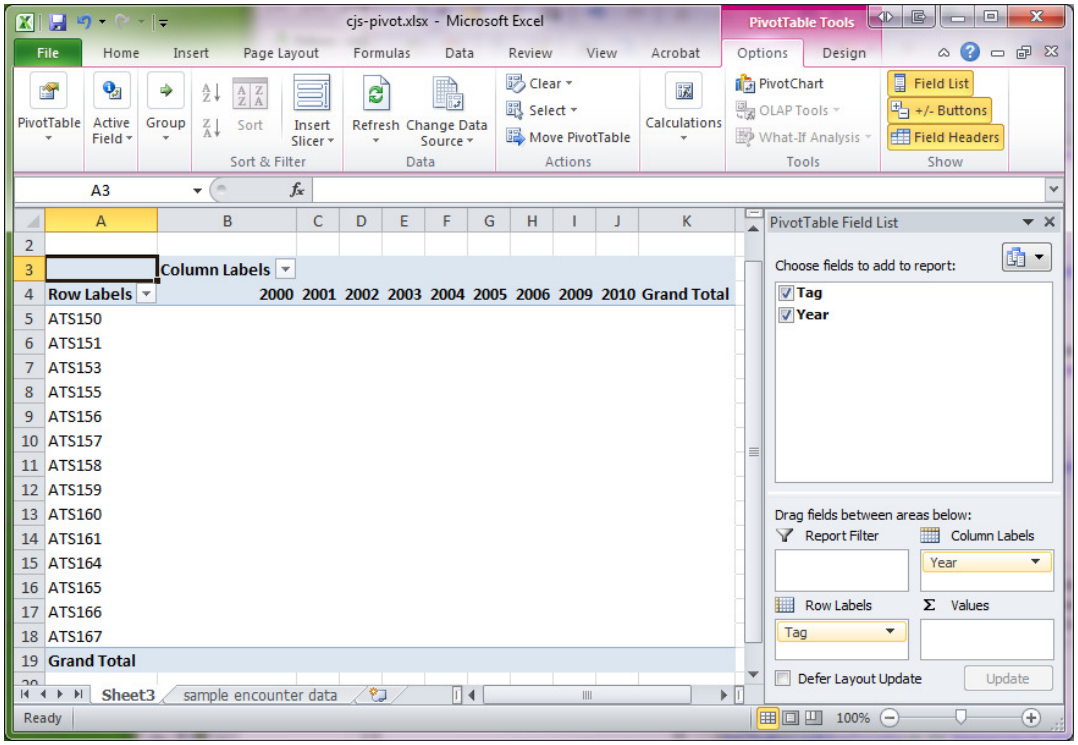
and then drag the '**`tag`**' field down to the '**`row labels`**' box at the bottom-right:



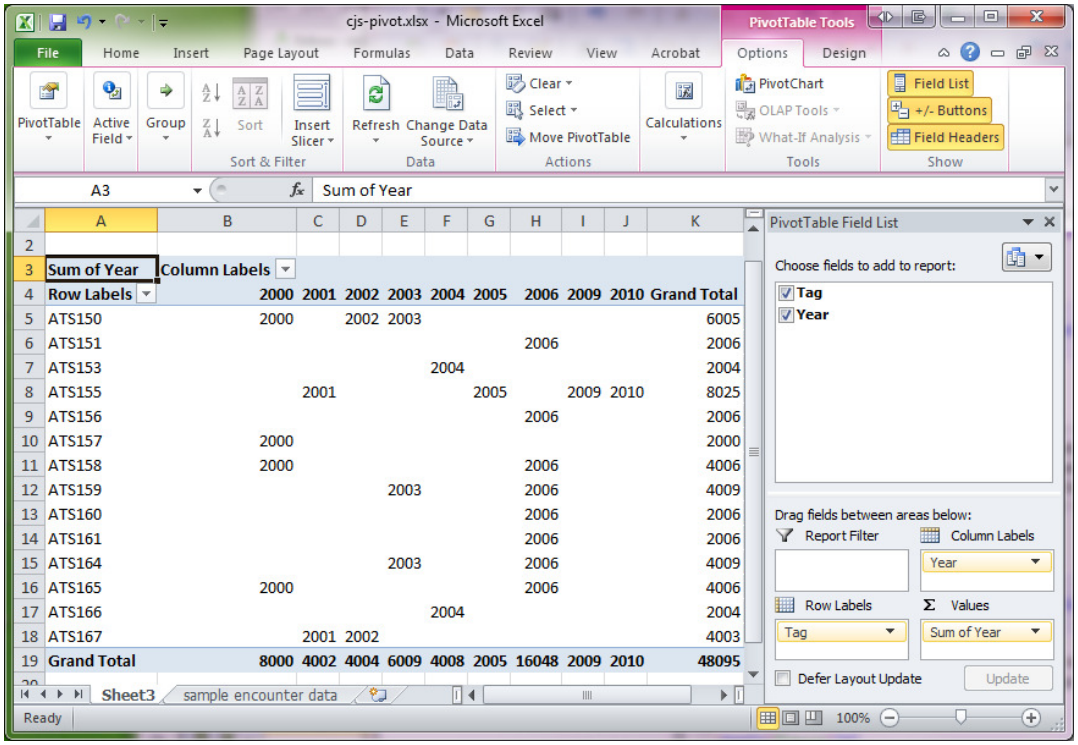
Then, do the same thing for the '**`Year`**' field: select '**`Year`**', and drag it down to the '**`column labels`**' box.



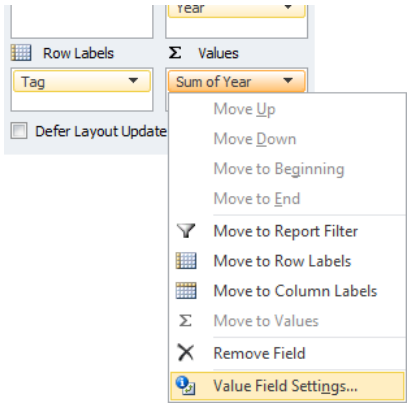
Once you have done this, you will quickly observe that a table (the 'pivot table') has been inserted into the main body of the template (see top of the next page). The table has row labels (individual tag numbers) and column labels (the years in your data file), plus some additional rows and columns for '**`Grand total`**' (reflecting the fact that pivot tables were designed primarily for business applications).



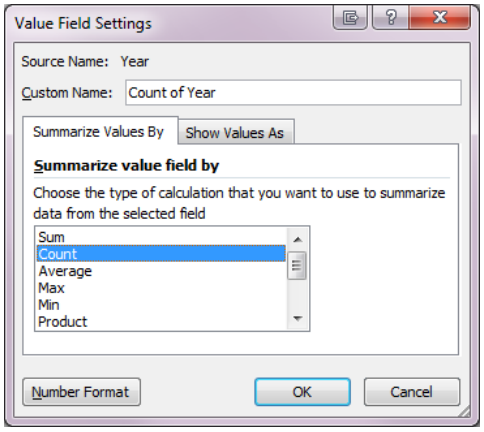
However, at present, there is nothing in the table (all of the cells are blank). Now, drag the '**year**' field label down to the '**values**' box in the lower right-hand corner.



What we see is that the year during which an encounter has occurred for a given individual has been entered explicitly into the table, in the column corresponding to that year. But, for an encounter history, we want a '1' to indicate the encounter year, not the year itself, and a '0' to indicate a year when an encounter did not occur. Achieving the first objective is easy. Simply pull down the '**Sum of Year**' menu, and select '**Value Field Settings...**':



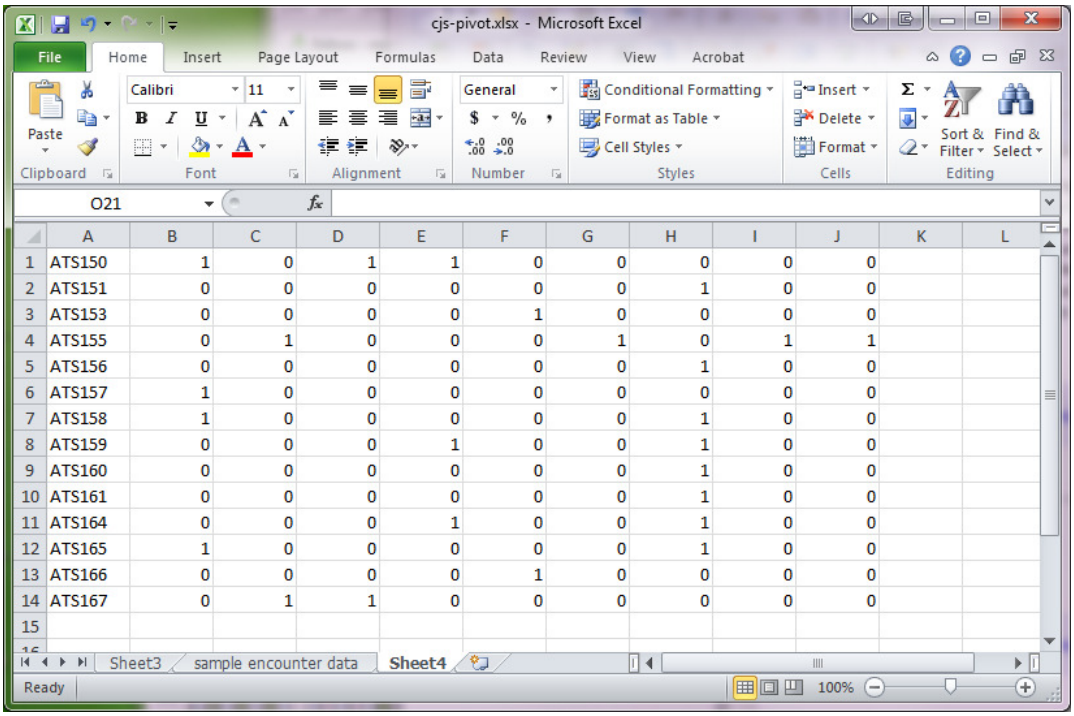
Then, switch the '**Summarize value field by**' selection from '**Sum**' to '**Count**':



As soon as you do this, then all of the years in the pivot table will be changed to 1. Why? Simple – all you've told the pivot table to do is count the number of times a year occurs in a given cell. Since the data file contains only a single record for each individual for each year it was encountered, then it makes sense that the tabulated '**Count**' should be a 1. Moreover, now the '**Grand Total**' rows and columns have some relevance - they indicate the number of individuals encountered in a given year (column totals), or the number of times a given individual was caught over the interval from 2000 to 2010 (row totals).

OK, on to the next step - putting a '0' in the blank cells for those years when an individual wasn't caught. This sounds easy enough in principle – a reasonable approach would be to select the rows and columns, and execute a 'search and replace', replacing blank cells with '0'. In fact, this is exactly what we want to do. However, for various reasons, you can't actually edit a pivot table. What you need to do first is select and copy the rows and columns (including the row labels, but excluding row and column totals), and paste them into a new worksheet. Then, simply do a '**Find & Select**'), replacing blanks

(simply leave the '**Find what**' field empty) with a '0'. The result is shown below:



	A	B	C	D	E	F	G	H	I	J	K	L
1	ATS150	1	0	1	1	0	0	0	0	0		
2	ATS151	0	0	0	0	0	0	1	0	0		
3	ATS153	0	0	0	0	1	0	0	0	0		
4	ATS155	0	1	0	0	0	1	0	1	1		
5	ATS156	0	0	0	0	0	0	1	0	0		
6	ATS157	1	0	0	0	0	0	0	0	0		
7	ATS158	1	0	0	0	0	0	1	0	0		
8	ATS159	0	0	0	1	0	0	1	0	0		
9	ATS160	0	0	0	0	0	0	1	0	0		
10	ATS161	0	0	0	0	0	0	1	0	0		
11	ATS164	0	0	0	1	0	0	1	0	0		
12	ATS165	1	0	0	0	0	0	1	0	0		
13	ATS166	0	0	0	0	1	0	0	0	0		
14	ATS167	0	1	1	0	0	0	0	0	0		
15												

(Alternatively, if you navigate to '**PivotTable | PivotTable Name | Options**', you will see an option to specify what an empty cell should show. Simply change it to a '0').

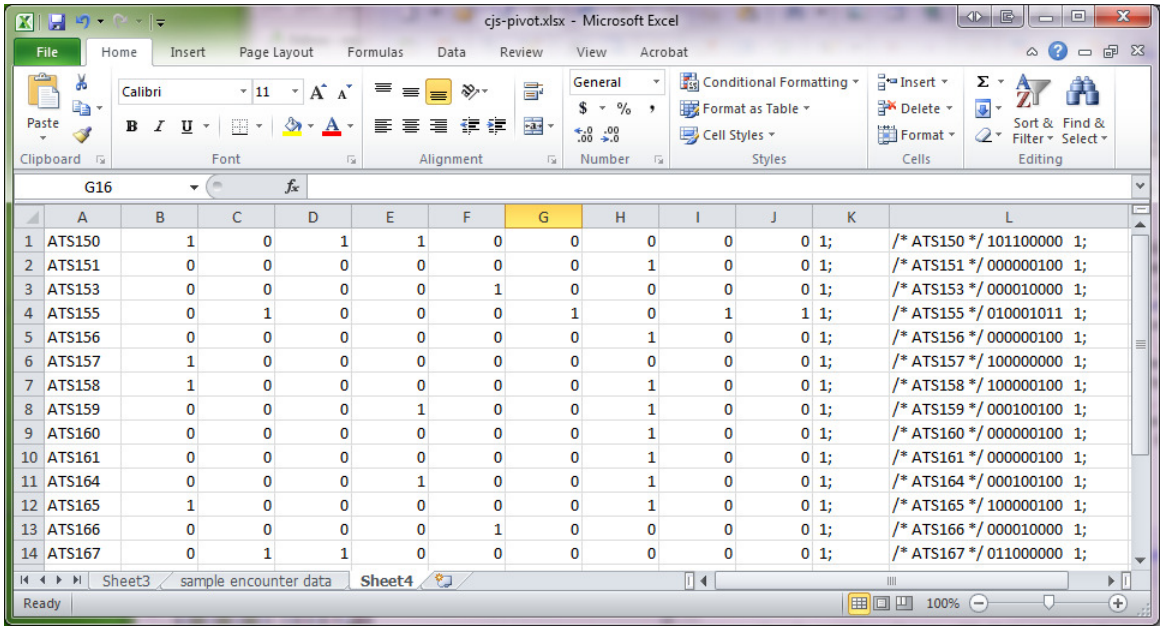
We're clearly getting closer. All that remains is to do the following. First, we remember that each line of the encounter history file must end with a frequency - where each line in the file corresponds to a single individual, then this frequency is simply '1;'. So, we simply enter '1;' into column **K**, and copy it down for as many rows as there are in the data (there are a number of ways to copy a value down a set of rows - we'll assume here you know of at least one way to do this).

Now, for a final step - we ultimately want an encounter history (.INP file) where the encounters form a contiguous string (i.e., no spaces). We can achieve this relatively easily by using the **CONCATENATE** function in Excel. Simply click the top-most cell in the next empty column (column **L** in our example), and then go up into the function box, and enter

```
=CONCATENATE("/* ",A1," */ ",B1,C1,D1,E1,F1,G1,H1,I1,J1,"  ",K1)
```

In other words, we want to 'concatenate' (merge together without spaces), various elements - some from within the spreadsheet, others explicitly entered (e.g., the delimiters for comments, so we can include the tag information, and some spacer elements).

Once you execute this cell macro, you can copy it down in column **L** over all rows in the file. If you manage to do this correctly, you will end up with a spreadsheet looking like the one shown at the top of the next page. All that remains is to select column **L** (which contains the formatted, concatenated encounter histories), and paste them into an ASCII text file. (A reminder here that you should avoid – as in 'like the plague' – using Word or Notepad as your ASCII editor. Do yourself a favor and get yourself a real ASCII editor. As mentioned earlier, there are a number of very good 'free' applications you can – and should – use instead of Notepad (e.g., Notepad++, EditPad Lite, jEdit, and so on...).



	A	B	C	D	E	F	G	H	I	J	K	L
1	ATS150	1	0	1	1	0	0	0	0	0	1;	/* ATS150 */ 101100000 1;
2	ATS151	0	0	0	0	0	0	1	0	0	1;	/* ATS151 */ 000000100 1;
3	ATS153	0	0	0	0	1	0	0	0	0	1;	/* ATS153 */ 000010000 1;
4	ATS155	0	1	0	0	0	1	0	1	1	1;	/* ATS155 */ 010001011 1;
5	ATS156	0	0	0	0	0	0	1	0	0	1;	/* ATS156 */ 000000100 1;
6	ATS157	1	0	0	0	0	0	0	0	0	1;	/* ATS157 */ 100000000 1;
7	ATS158	1	0	0	0	0	0	1	0	0	1;	/* ATS158 */ 100000100 1;
8	ATS159	0	0	0	1	0	0	1	0	0	1;	/* ATS159 */ 000100100 1;
9	ATS160	0	0	0	0	0	0	1	0	0	1;	/* ATS160 */ 000000100 1;
10	ATS161	0	0	0	0	0	0	1	0	0	1;	/* ATS161 */ 000000100 1;
11	ATS164	0	0	0	1	0	0	1	0	0	1;	/* ATS164 */ 000100100 1;
12	ATS165	1	0	0	0	0	0	1	0	0	1;	/* ATS165 */ 100000100 1;
13	ATS166	0	0	0	0	1	0	0	0	0	1;	/* ATS166 */ 000010000 1;
14	ATS167	0	1	1	0	0	0	0	0	0	1;	/* ATS167 */ 011000000 1;

Other data types

Here we will consider 2 other data types, the robust design, and multi-state. Clearly, there are more data types in **MARK**, but these two represent very common data types, and if you understand steps in formatting `.INP` files for these two data types, you'll more than likely be able to figure out other data types on your own.

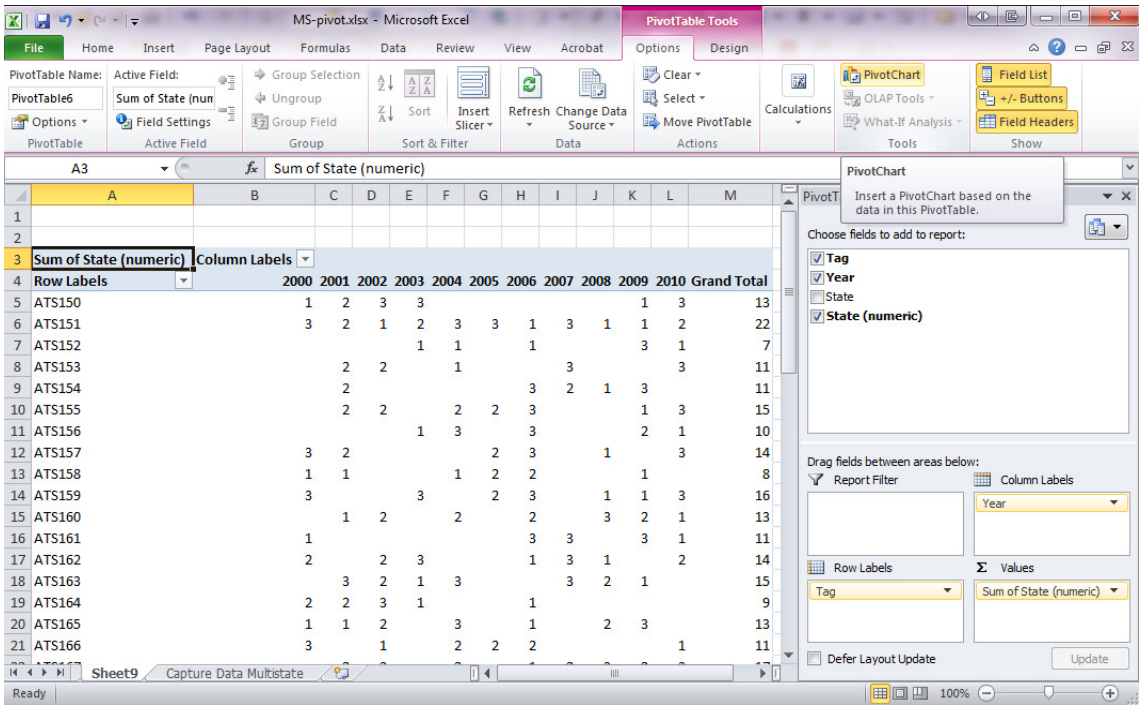
multi-state

Here we will demonstrate formatting an `.INP` file for a multi-state data set (see Chapter 10). The encounter data we will use are contained in the Excel spreadsheet `MS-pivot.xlsx`. The file consists of 3 columns: `TAG` (indicating the individual), `YEAR` (the year the individual was encountered), and the `STATE` (for this example, there are 3 possible states: `F`, `U`, `N`).

We start by noting that `STATE` is a character (i.e., a letter). This might seem perfectly reasonable, since the most appropriate state name (indicator) might be a character. Unfortunately, Excel can't handle characters in the table cells when you pivot the table. As such, you first need to (i) select the column containing the state variable, (ii) copy this into the first empty column, and (iii) execute a '**`Find and Replace`**' in this column, such that you change `F` → `1`, `U` → `2`, and `N` → `3`. Once finished, your Excel spreadsheet shoot look something like what is shown to the right.

	A	B	C	D
1	Tag	Year	State	State (numeric
2	ATS150	2000	F	1
3	ATS150	2001	U	2
4	ATS150	2002	N	3
5	ATS150	2003	N	3
6	ATS150	2009	F	1
7	ATS150	2010	N	3
8	ATS151	2000	N	3
9	ATS151	2001	U	2
10	ATS151	2002	F	1
11	ATS151	2003	U	2
12	ATS151	2004	N	3
13	ATS151	2005	N	3
14	ATS151	2006	F	1
15	ATS151	2007	N	3

Next, select the data, and inset a Pivot Table into a new sheet in the spreadsheet. Drag `TAG` to the '**`Row Labels`**' box, `YEAR` to the '**`Column Labels`**' box, and `State (numeric)` to the '**`Values`**' box, as shown below.

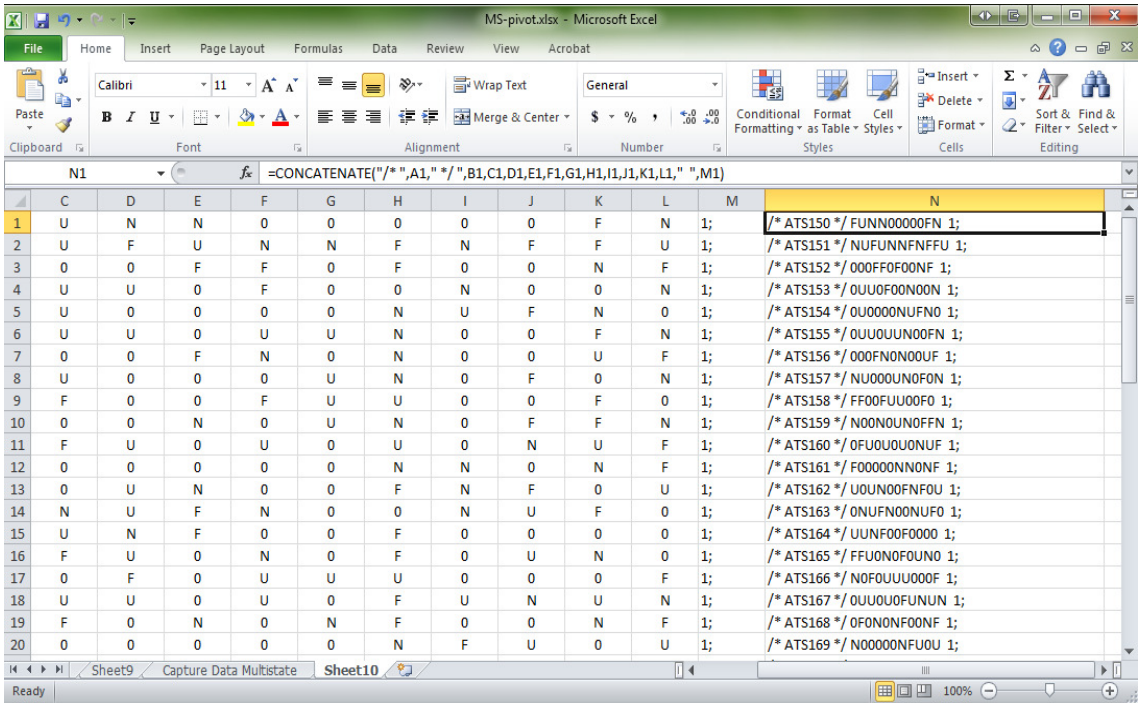


Next, copy the `TAGS`, `YEARS` and table values to a new worksheet. Then '**`Find and Replace`**' all the blank cells with zeros. At this point, you have a decision to make: you can either (i) '**`Find and Replace`**' the states from numeric back to their original character values (i.e., 1 → `F`, 2 → `U` and 3 → `N`), or (ii) leave the states numeric, and simply inform **MARK** what the states mean. For this example, we'll '**`Find and Replace`**' the states from numeric back to their original character values. Finally, add a column of '`1;`' to the new worksheet.

Then click the top-most cell in the next empty column (column **L** in our example), and then go up into the function box, and enter

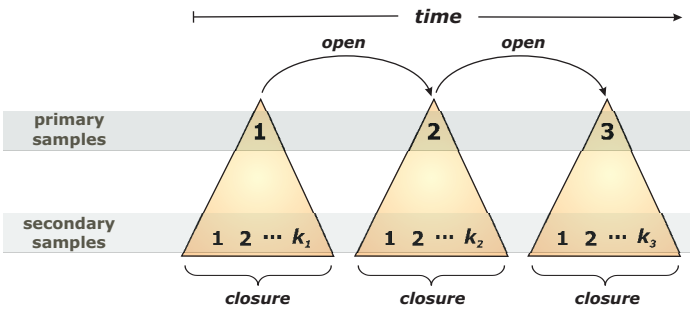
```
=CONCATENATE("/* ",A1," */ ",B1,C1,D1,E1,F1,G1,H1,I1,J1,K1,L1,"  ",M1)
```

In other words, we want to 'concatenate' (merge together without spaces), various elements - some from within the spreadsheet, others explicitly entered (e.g., the delimiters for comments, so we can include the tag information, and some spacer elements). Once you execute this cell macro, you can copy it down in column **L** over all rows in the file. The final worksheet should look something like the one shown at the top of the next page. At this point, you simply copy your concatenated encounter histories from column **N** into an editor, and save into an `.INP` file.



robust design

For our final example, we consider formatting an `.INP` file for a robust design analysis (the robust design is covered in Chapter 15). In brief, the robust design combines closed population samples embedded (nested) within open population samples. Consider the figure at the top of the next page. As shown, there are 3 'open population' samples (known as primary period samples). Between open samples, population abundance can change due to emigration, death, immigration or birth. Within each open sample period are embedded k 'closed population' (or secondary) samples. The trick here is to encode the encounter history taking into account the presence of both primary and secondary samples (where the number of secondary samples may vary among primary samples). As you might expect, the greater complexity of the RD encounter file might require a somewhat higher level of Excel proficiency than the first two examples we discussed earlier.



In this example (data contained in `RD-pivot.xlsx`), we assume primary samples from 2000-2010. Within each primary period, we have 4 secondary samples, which occur from May 1 to May 15 (sec-

ondary sample 1), May 16 to May 30 (secondary sample 2), June 1 to June 15 (secondary sample 3), and June 16 to June 30 (secondary sample 4). For each secondary sample, and encountered individual is recorded only once. We imagine that your data are stored in the following way. For each individual (`TAG`), for each primary sample (`YEAR`), you have a series of columns, one for each secondary sampling period.

	A	B	C	D	E	F	G
1	Tag	Date	Year				
2							
3	ATS150		2000		5/30/12		6/17/12
4	ATS150		2001	5/2/12			6/24/12
5	ATS150		2002			6/4/12	
6	ATS150		2003		5/23/12	6/2/12	6/25/12
7	ATS150		2009				
8	ATS150		2010	5/6/12	5/17/12	6/10/12	6/17/12
9	ATS151		2000	5/13/12	5/18/12		
10	ATS151		2001	5/7/12			
11	ATS151		2002			6/10/12	6/18/12
12	ATS151		2003			6/1/12	6/16/12
13	ATS151		2004	5/6/12	5/25/12	6/10/12	6/19/12
14	ATS151		2005	5/11/12	5/18/12		
15	ATS151		2006	5/11/12	5/28/12		
16	ATS151		2007	5/9/12			
17	ATS151		2008	5/2/12			
18	ATS151		2009				6/23/12
19	ATS151		2010			6/12/12	6/25/12
20	ATS152		2003	5/12/12	5/25/12	6/3/12	6/29/12

For example, in the preceding figure, we see that individual with tag '`ATS150`' was observed during primary sample, 2000, 2001, 2002, 2003, 2009, and 2010. In 2000, the individual was not observed during the first secondary sample (May 1 to May 15), was observed during the second secondary sample (May 16 to May 30), was not observed during the third secondary sample (June 1 to June 15), and was observed during the fourth and final secondary sample (June 16 to June 30). In contrast, in 2010, the individual with tag '`ATS1150`' was observed in all 4 secondary samples.

Now, you may be wondering why we've entered dates in terms of 2012, even for primary encounter years <2012. For example, for '`ATS150`', we enter '`5/30/12`' as the date for the encounter during the second secondary sample period. We need to do this in order to make use of some very handy Excel functions. For example, consider the '`year`' function. This function extracts the year associated with a given date (such that if you type in '`=year(B2)`' and `B2` is a date, it will return the year associated with that date. So, for robust design data, you may have intervals (for a secondary sample period) spanning from 5/1/12 to 5/15/12, and you want to know if the encounter date falls between them.

All you need to do is

- use the `AND` function to determine if a date falls within a given range. For example, in cell `H3` in the spreadsheet, we enter

  ```
  =AND(D3>=H1,D#<=H2)
  ```

- What you are asking Excel is: "Is `D3` (my date of capture) greater than or equal to my first date, `5/1/12`, and less than or equal to `5/15/12`". We do the same thing for each of the other 3 secondary sample periods.
- This may seem a bit odd at first but keep in mind that Excel treats all dates as a number of days since January 1, 1900 or 1904 (depending on which version of Excel you are using)
- The `AND` function will return a `TRUE` value is the criteria in the parenthesis are met or a `FALSE` value if they are not

- Once you have got all of your `TRUE` and `FALSE` values copy them into a separate set of columns. Note that instead of just '**`paste`**'or '**`ctrl+v`**', you want to right click and '**`paste special`**' and select the '**`Values`**' box. This tells Excel to just give you the displayed number text or whatever appears in the box without any of the underlying formulas.
- Now you can '**`Find and Replace`**' `TRUE` with `1` and `FALSE` with `0`

These steps (and cell macros) are shown in worksheet '`RD within season period trick`'. At this point, you will se something that look like

I10 =AND(E10>=$I$1,E10<=$I$2)

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	Tag	Date	Year					5/1/2012	5/16/2012	6/1/2012	6/16/2012		Interval 1	Interval 2	Interval 3	Interval 4
2								5/15/2012	5/30/2012	6/15/2012	6/30/2012					
3	ATS150		2000		5/30/12		6/17/12	FALSE	TRUE	FALSE	TRUE		0	1	0	1
4	ATS150		2001	5/2/12			6/24/12	TRUE	FALSE	FALSE	TRUE		1	0	0	1
5	ATS150		2002			6/4/12		FALSE	FALSE	TRUE	FALSE		0	0	1	0
6	ATS150		2003		5/23/12	6/2/12	6/25/12	FALSE	TRUE	TRUE	TRUE		0	1	1	1
7	ATS150		2009					FALSE	FALSE	FALSE	FALSE		0	0	0	0
8	ATS150		2010	5/6/12	5/17/12	6/10/12	6/17/12	TRUE	TRUE	TRUE	TRUE		1	1	1	1
9	ATS151		2000	5/13/12	5/18/12			TRUE	TRUE	FALSE	FALSE		1	1	0	0
10	ATS151		2001	5/7/12				TRUE	FALSE	FALSE	FALSE		1	0	0	0
11	ATS151		2002			6/10/12	6/18/12	FALSE	FALSE	TRUE	TRUE		0	0	1	1
12	ATS151		2003			6/1/12	6/16/12	FALSE	FALSE	TRUE	TRUE		0	0	1	1
13	ATS151		2004	5/6/12	5/25/12	6/10/12	6/19/12	TRUE	TRUE	TRUE	TRUE		1	1	1	1
14	ATS151		2005	5/11/12	5/18/12			TRUE	TRUE	FALSE	FALSE		1	1	0	0
15	ATS151		2006	5/11/12	5/28/12			TRUE	TRUE	FALSE	FALSE		1	1	0	0
16	ATS151		2007	5/9/12				TRUE	FALSE	FALSE	FALSE		1	0	0	0
17	ATS151		2008	5/2/12				TRUE	FALSE	FALSE	FALSE		1	0	0	0
18	ATS151		2009				6/23/12	FALSE	FALSE	FALSE	TRUE		0	0	0	1
19	ATS151		2010			6/12/12	6/25/12	FALSE	FALSE	TRUE	TRUE		0	0	1	1

At this point, the remaining steps are similar to the same steps we used for CJS and MS data types (as described earlier). You simply

1. copy the the data to a new worksheet (shown in '`capture data-robust design`')
2. Select the data, then '**`Insert | Pivot Table | Pivot Table`**'
3. Drag `Tag` to '**`Row Label`**', `Year` to '**`Column Label`**'
4. Now here is another difference for the RD: there are multiple occasions per year. So just drag each one to the values box in the order that they occur!
5. concatenate into a contiguous encounter history, and you're done. Have a look at the worksheet '`RD Input Construction`' for what it should look like.

CHAPTER 3

First steps in using Program MARK. . .

In this chapter we will introduce the basic mechanics of running program **MARK**, using a small data set consisting of 7 years of capture-recapture data on a small passerine bird, the European Dipper (*Cinclus cinclus*). This data set is the same as that used in 'Examples' in Lebreton *et al*. (1992), and consists of marking and recaptures of 294 breeding adults each year during the breeding period, from early March to 1 June. All birds in the sample were at least 1 year old when initially banded. We'll forgo discussion of GOF (Goodness of Fit) testing for the moment, although we emphasize that, in fact, this is the prerequisite step before you analyze your data. GOF testing is covered later.

Our main intent here is to show you the basics of running **MARK**, not to provide great detail on 'why you are doing what you are doing'. Our experience has shown that perhaps the greatest initial hurdle to using a new piece of software, especially one as sophisticated as **MARK**, is the 'newness' of the interface, and sheer number of options available to the user. In addition, we are starting with the Dipper data set, since it has been extensively analyzed in several places, and will be very familiar to many experienced users migrating to **MARK** from another application.

We also believe that starting with a 'typical' mark-recapture problem is a good place to begin - if you understand how to do a mark-recapture analysis, you're much of the way to understanding the principles behind many of the other analytical models incorporated into **MARK**. The male subset of the Dipper data set is not one of those which are 'bundled' with **MARK** (although the full dipper data set is). We'll assume here that you've managed to extract the Dipper data set `ED_MALES.INP` from the `markdata.zip` file that accompanies this book (if you don't have `markdata.zip`, you can download it from the same website you downloaded this book from).

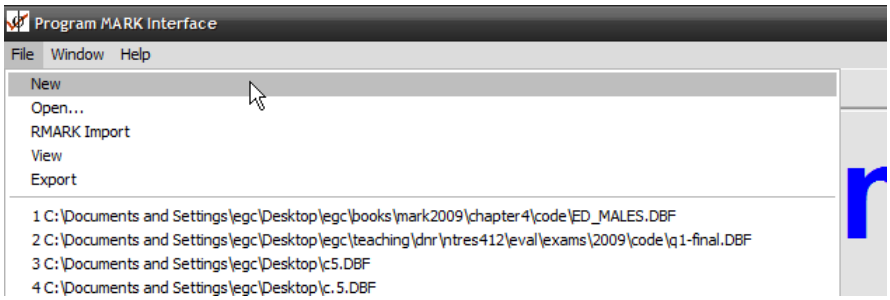
3.1. Starting MARK

Since **MARK** is a true Windows application, starting it is as simple as double-clicking the **MARK** icon (which we assume resides in some specified folder on your desktop). Locate the icon, and double-click on it. **MARK** is a fairly large program (although this is now a relative statement with the advent of 100+ MB word processors!), and may take a few moments to start up. If all goes well, you should soon be presented with the opening 'splash screen' (shown at the top of the next page – the particular 'warm and fuzzy organism' you see will depend on which version of **MARK** you are using), indicating that **MARK** is 'up and running'. If nothing happens (or you get some typically obscure Windows error message), this is a good indication that something is not working right. Unfortunately, figuring out the problem depends to a large degree on things like: how many other applications do you have open? Are you sure you've installed the most recent version of **MARK**? How much memory is in your machine? Are

you 'lucky' enough to still be running Windows Vista? And so on, and so on. **MARK** is very robust on most machines, so problems getting it started should be very infrequent. If it doesn't start correctly, then try again after first closing all other running applications (in general, **MARK** runs 'best' when it is the only program running). If that doesn't work, then trying re-installing from scratch - if you've already downloaded the `setup.exe` file, then this should only take a few moments (again, when in doubt, try reinstalling - this is often a good way to 'fix' minor problems that might arise).

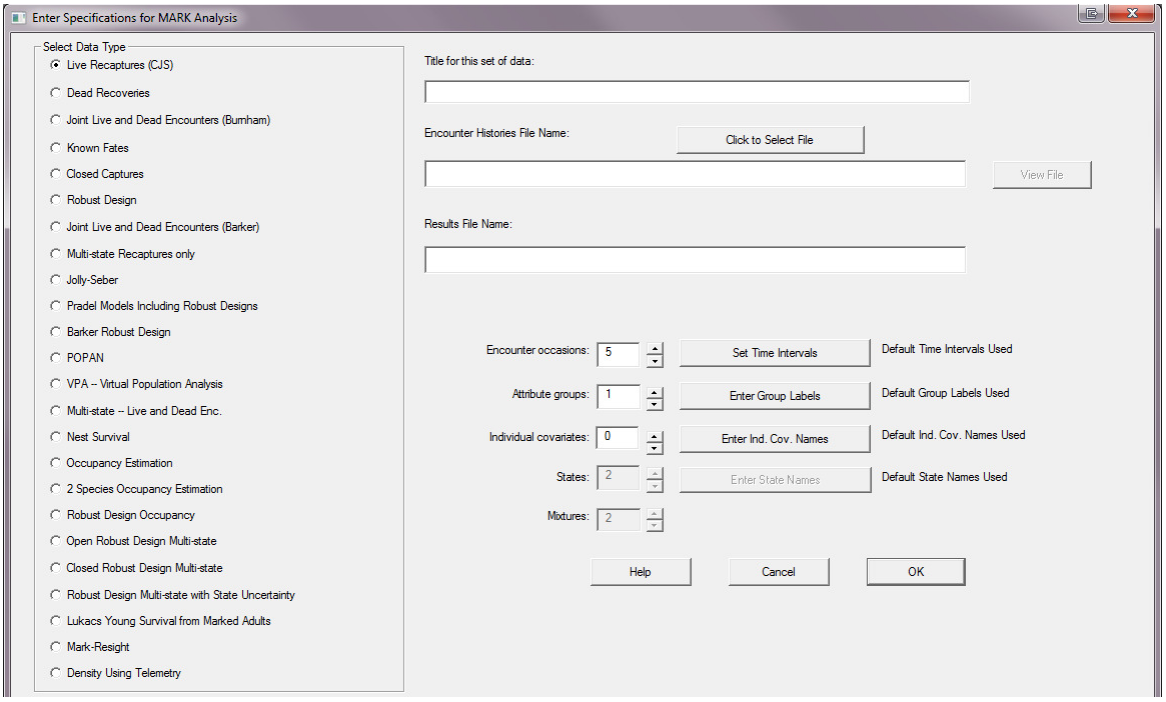
3.2. starting a new project

The very first thing you need to do is to tell **MARK** you're going to start a new project (we're assuming that at this stage, you don't have any existing **MARK** projects going). Doing this is very easy - all you need to do is pull down the '**`File`**' menu in the upper left-hand corner of the main **MARK** window, and select '**`New`**' from the drop-down menu:



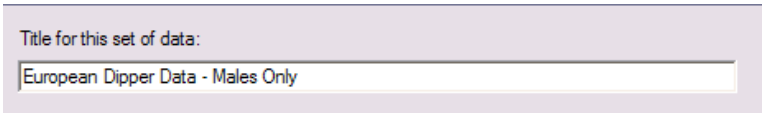
Once you have selected, and then released the '**`New`**' option from the drop-down '**`File`**' menu, the

graphical 'splash-screen' that you saw when you started **MARK** will be erased, and you will be presented with a new sub-window – the specification window for program **MARK**:



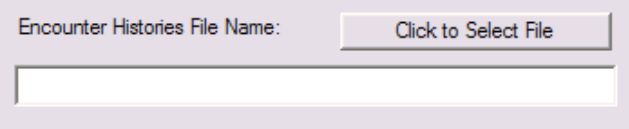
Clearly, the specification window contains a fair bit of information, but the 'basics' can be broken down into 4 main sections.

First, on the left of the window (highlighted by a bezeled line) is a simple radio-button list for the various data types **MARK** can handle. In fact, this is the point at which you tell **MARK** what kind of analysis you want to do. In its simplicity, this list belies the sheer scope of analytical coverage provided by **MARK**. Whereas most previous applications specialized on one (or a couple) of types of analysis (for example, live recapture-only, or dead recovery-only), **MARK** can handle most of the common analytical designs in use today. While **MARK** is clearly not a replacement (in some ways) for a more general purpose approach like **SURVIV** (or, more recently, **R**, **MATLAB** or **WinBUGS**), it **is** in many respects a replacement for much (if not all) of the 'canned' software previously in general use. The '**`live recaptures (CJS)`**' data type is selected by default - and, is the data type we want for this analysis. At the top of the right-hand half of the window is a fill-in element for the '**`Title`**' of the project. This is available as a convenience to the user, and does not have to be filled out. For this example, we use a title reflecting the fact we're analyzing the male data from the European Dipper study.

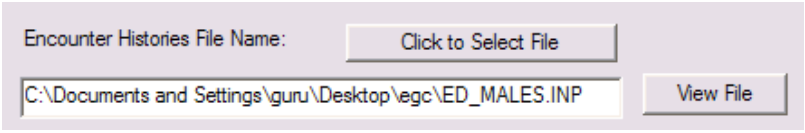


Immediately below the title field is a second fill-in element where the user specifies the file containing the encounter histories they want to analyze. If the file name and path are known, they can be entered directly into this box. More typically, you will want to browse for the file (i.e, select it manually), by

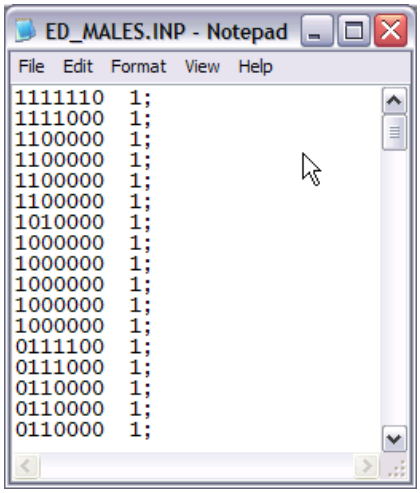
clicking on the '**Click to Select File**' button immediately below the box (you can also double-click the fill-in box itself). If you click on this button, you will be presented with the standard Windows interface to finding and selecting a particular file (see below).



One thing to note is that until you have selected a file to analyze, the '**View File**' button is not available (i.e., is not active). Once you have selected the file, you will be able to view its contents. This is useful if you forget certain things about your data (for example, the number of occasions). In our example, we select the file `ED_MALES.INP` (obviously, the path shown here reflects the machine we're running **MARK** on, and will **not** be the same as what might appear on your machine).



Again, note that once the file has been specified, the '**View File**' button become active. If we click on **View File**, **MARK** starts up the default Windows browser (usually the Windows **Notepad** - if your file is too large for **Notepad** to handle, then Windows will prompt you to use an alternative application to view the file). We see from the **Notepad** window (pictured below) that we have 7 occasions, and only 1 group (recall from Chapter 2 the basics of formatting data for processing by program **MARK**). You might want to note that since this is the Windows **Notepad**, you can, if necessary, edit the input file at this stage.

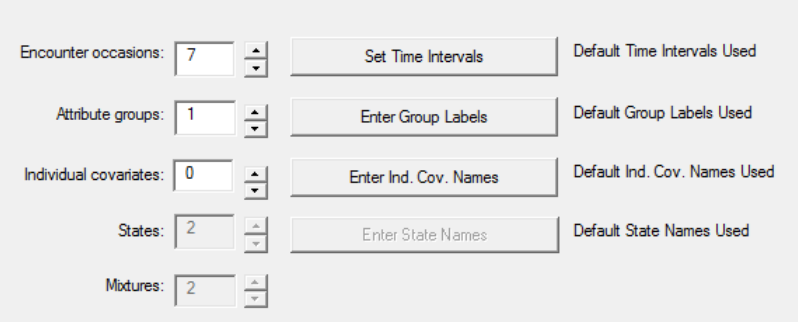


Finally, the bottom-half of the right hand side. Here, you specify the number of occasions in the file. Because the `.inp` files containing the capture histories do not explicitly code for the number of capture histories in the file, you will have to tell **MARK** how many occasions there are. It defaults to 5 – do **not**

be fooled into thinking this is the number of occasions in your file. **MARK** has no way of knowing what this value is - you have to enter it explicitly. You can also tell **MARK** how many '**attribute groups**' are in the data (for example, if your file contained data for both males and females, you would enter 2). Finally, the number of individual covariates. (*Note*: the boxes for the '**number of strata**' and **Mixtures**' only becomes available if you select the appropriate data types which require specifying these options.)

To the right of each input box, there is a button which allows you to control some aspects of each box. For example, to the right of the number of occasions box is a button which lets you set the intervals between each occasion. The default is '**1 time period between each occasion**'. We'll talk more about this particular option later. The other two buttons for '**group labels**' and '**individual covariate**' names are (we suspect) fairly self-explanatory. For this analysis, since we have only one attribute group (males), there is no real need to specify a particular label.

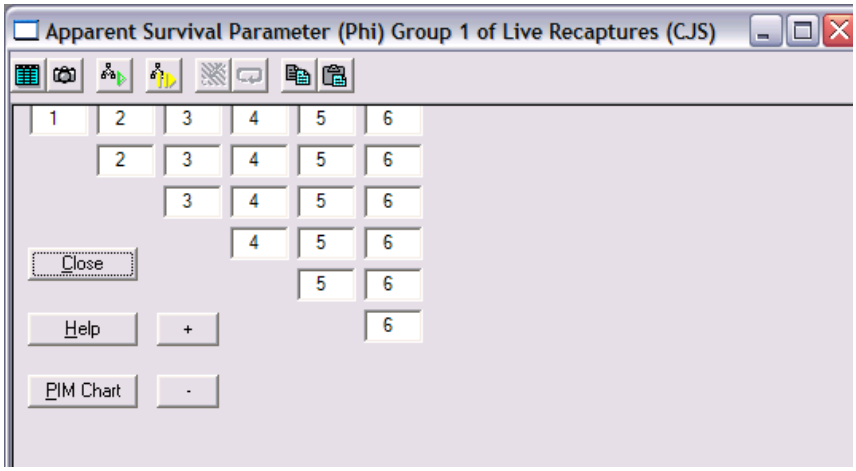
In our Dipper example, we have 7 occasions, and only 1 group (as shown below):



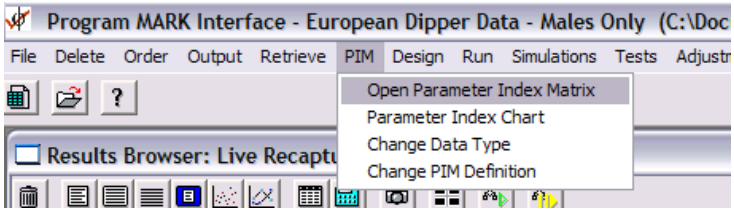
Now, once you've got everything set in the specification window, you're ready to proceed (if not, simply correct the individual entries as needed, or cancel to start over). To continue, simply click the '**OK**' button.

The first thing that will happen when you press the '**OK**' button (assuming that you've specified a file that exists) is that **MARK** will close the specification window, and present you with a small little pop-up window telling you that it has created a `DBF` (data-base format) file to hold the results of your analyses. The name of the file in this example, is `ED_MALES.DBF`. **MARK** uses the prefix of the `.inp` data file (in this case, `ED_MALES`) as the prefix of the DBF file (resulting in `ED_MALES.DBF`). **MARK** pauses until you press the '**OK**' button, telling it to proceed. If `ED_MALES.DBF` already existed (i.e., if you'd already done some analysis on these data), **MARK** would inform you that it will have to overwrite the existing file, and will then ask you if this is OK.

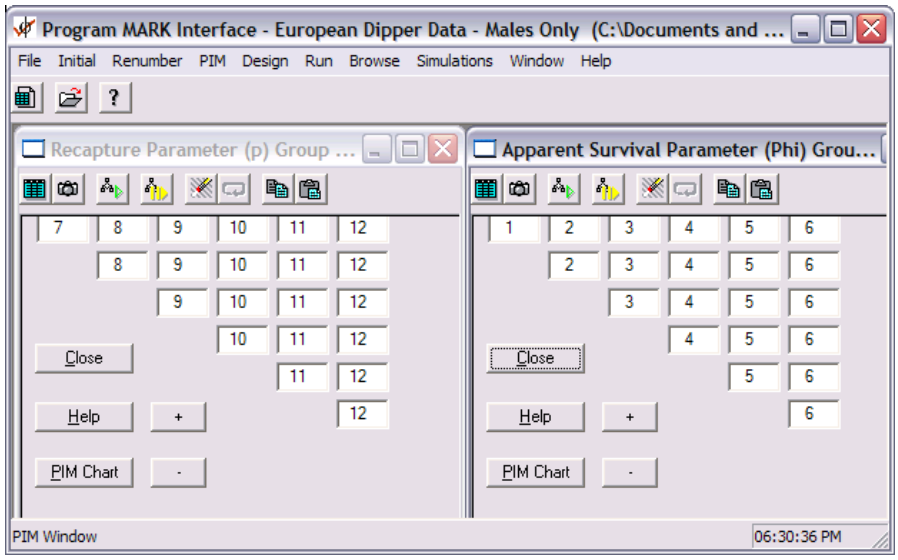
Since we haven't run any analysis on these data before (we're assuming here that this is your first attempt to use **MARK**!), we simply click the '**OK**' button. Doing so causes the pop-up window to close, and a window containing a 'triangular' matrix representation of the survival model structure is presented (shown at the top of the next page).



However, you'll note from the title of the window that the matrix represents only the survival parameters. For a mark-recapture analysis, we're also interested in estimating the recapture probability. What you don't see here is that, by default, **MARK** initially presents you with only the survival parameters, assuming (presuming?) that this is what you're most interested in working with. However, clearly we may (and probably should) also be interested in modelling the other parameters which define the system (in this case, the recapture parameter). For now, let's get **MARK** to show us both parameter matrices. We get **MARK** to show us any (or all) of the parameter matrices by accessing the '**`Parameter Information (or Index) Matrix`**' menu (PIM), and selecting the '**`Open Parameter Index Matrix`**' option:



Once you select this option, you will be presented with a new window containing a list of parameters which you can 'open' - in other words, a list of parameters you can ask **MARK** to show you in the 'triangular matrix' format. In our example, since the survival parameter chart is already open, the only one which will show up on the list is the recapture parameter. You can either select it individually, or use the '**`Select All`**' button. Once you have selected the recapture parameter matrix, you simply click '**`OK`**', to cause **MARK** to place the recapture PIM in its own window. Initially, they may (or likely will) be presented in an overlapped way (technically known in Windows-jargon as 'cascaded'). To see the 2 PIM windows separately, you can access the '**`Window`**' menu, and select '**`Tile`**'. This will cause the PIM windows to be arranged in a 'tiled' (i.e., non-overlapping) fashion.



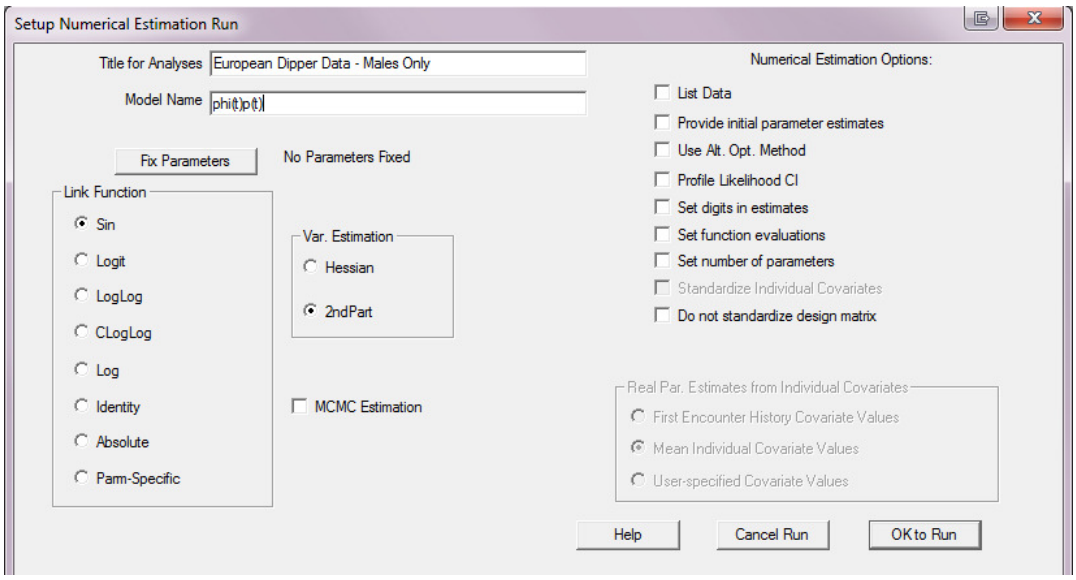
If you've had some background in mark-recapture analysis, you might recognize that these PIM's reflect the fully time-dependent Cormack-Jolly-Seber (CJS) model. If you're new to mark-recapture analysis, not to worry - all of this gets covered in great detail in subsequent chapters. For the moment, all you need to know is that in this analysis, the survival parameters are numbered 1 to 6 (there are seven occasions, so six intervals over which survival is estimated), and 6 recapture parameters (numbered 7 through 12). In fact, there are only 10 separable parameters and one 'product' parameter estimated in this analysis (11 total parameters), but we'll discuss the details concerning this later.

3.3. Running the analysis

Let's proceed to run the analysis. Not surprisingly, to run an analysis in **MARK**, you need to access the '**Run**' menu. In this example (and at this stage), when you access the '**Run**' menu, **MARK** will present you with several options - for now, we want to select the '**Current Model**' option.

The first thing that **MARK** does is present you with a new window - the '**Setup Numerical Estimation Run**' window (since this is a rather awkward window name, we'll refer to it simply as the '**Run**' window). Much like the specification window we saw earlier, the '**Run**' window (shown on the next page) contains a lot of options, so let's take some time at this point to get you familiar with the components of this window. As with the specification window, the run window can be broken down into 4 major elements. First, in the top left, the analysis title (which **MARK** carries over from what you entered earlier in the specification window), and the model name (which is initially blank). **MARK** must have a model title to refer to (the reason why will be obvious later), so we need to enter something in the model name input box. Following the naming convention advocated in Lebreton *et al*. (1992) (which follows closely from standard linear models notation), we'll use '`Phi(t)p(t)`' as the name for this model, reflecting the fact that the model we're fitting has full time-dependence for both survival (φ - since **MARK** doesn't allow you to enter non-ASCII text, we're restricted to writing out the Greek symbol as '`phi`') and recapture (p). The (t)-notation simply indicates time-dependence.

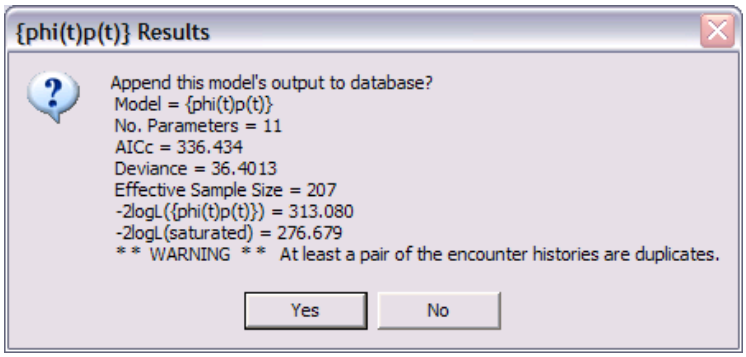
Next, the lower-left hand side of the run window. This is where we specify the link function (**MARK** lets you choose among 6 - the sin link is the default, while the logit link is perhaps the most familiar). We'll talk about link functions in more detail later. For the moment, we'll use the default sin link.



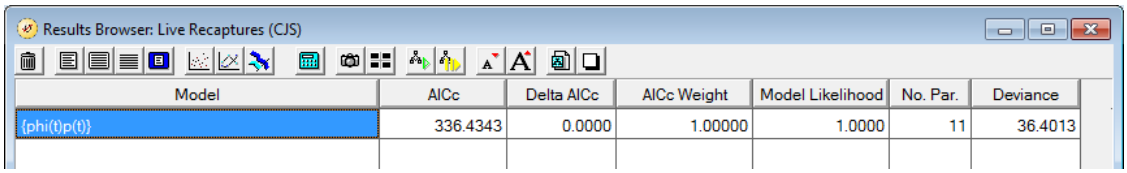
To the right of this list of link functions is a list of two '**Var. Estimation**' options - these are options which controls how the variance-covariance matrix is estimated. This is important because this estimation provides both the information needed to compute the standard errors of the estimates, but is also used to calculate the number of estimable parameters in the model (in this example, there should be 11 estimable parameters). The '**2nd Part**' option is the default and preferred option, so we'll use it here. Finally, just above the link and variance estimation lists is a button which allows you to '**fix parameters**'. As we'll see in subsequent chapters, there will be occasions when we need to specify a value (normally 0 or 1) for some parameters - in this example, there is no need to do so. Fixing parameters that are logically constrained to be a particular value is useful, since it both reduces the number of estimable parameters, and helps **MARK** estimate the remaining parameters more accurately and efficiently.

On the right-hand side of the run window are a series of program options related to the numerical optimization. Most of the options are fairly self-explanatory, so we won't go into any of these in detail here. For now, we'll leave all of the check boxes on the right-hand side blank.

You're now ready to run the program. To do this, simply click on the '**OK to run**' button in the lower right-hand corner of the run window. If you are using the default preferences for **MARK** (which you probably will be at this point), **MARK** will spawn a box asking you if you want to use the 'identity matrix' (since none was specified). The use of the identity matrix will be covered in much more detail in Chapter 6 - for now, simply accept the 'identity matrix', by clicking '**Yes**'. Immediately after doing so, **MARK** spawns a numerical estimation window (black background). This window, which is a shell to the numerical estimation routines in **MARK**, allows you to watch the progress of the estimation. Several things will scroll by pertaining to the estimation. Much of the information being printed to the estimation window can be also printed out to the results DBF, by specifying the appropriate options in the run window. You may also notice that **MARK** has created another window, called the '**Results Browser**' - more on this window in a moment. Once the estimation is complete (which should only take a second or two for this dataset), **MARK** will shut down this command window, and then present you with the 'pop-up' window shown at the top of the next page.



This window provides summary of some of the key results of your estimation (the number of parameters estimated, the model deviance and so forth). The purpose of this summary is to give you the option of whether or not to append the full results of the estimation to the result database file (`DBF`). In theory, you could decide at this point that there was something 'wrong', and not append the results. You might wonder about the `**WARNING**` statement at the bottom - if you read it carefully, you'll see that **MARK** is 'warning' you that at least one pair of encounter histories in the `.INP` file are duplicates. In fact, for `.INP` files where each line represents the encounter history for a different individual (as in the present example), this is entirely expected. Meaning, there is nothing to worry about in this case. So we click '**Yes**', and accept the results. As soon as you click '**Yes**', two things happen in rapid succession. First, the results summary window closes. Next, an item is added to the results browser window.

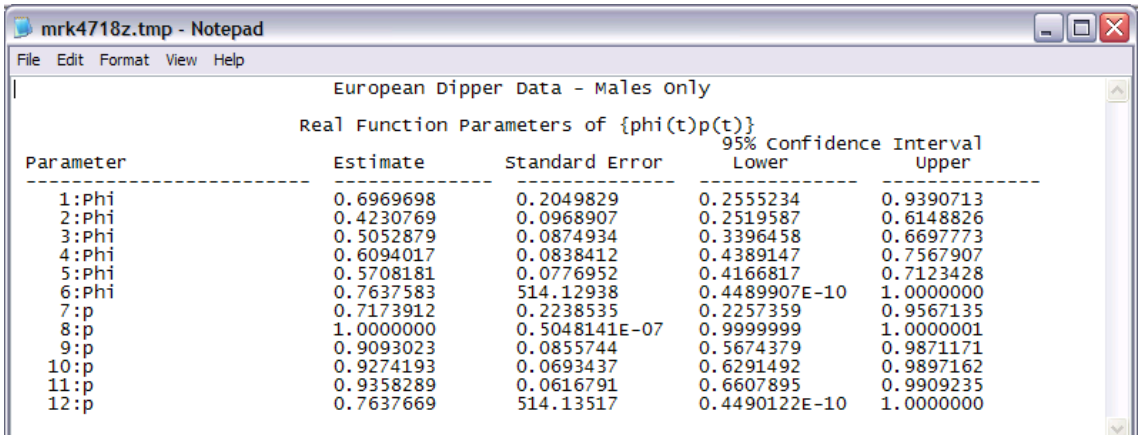


3.4. Examining the results

Now that we have something to 'look at' in the browser window, we can focus a bit more on the structure of the window, and what options are available. First, in the main body of the results browser, you have 5 columns of information. From left to right: the model title (or simply, '**Model**'), the corrected or adjusted Akaike's Information Criterion (AIC_c), the ΔAIC value (the difference in the value of the AIC from the model currently in the browser having the lowest AIC – since we have only one model in the browser at the moment, $\Delta AIC = 0$), the AIC weight, the model likelihood, the number of estimated parameters, and the model deviance. The column arrangement can be modified to suit your preferences by simply dragging the columns to the left or right. The AIC_c, the number of parameters and the model deviance (in particular) form the basis for comparison with other models. Since we only have one model at the moment, we'll defer discussion of these issues until later. Along the top of the browser window are a number of buttons, which you can use to access a variety of functions. More on these later.

For now, we want to view the results of our estimation. We can view just the reconstituted parameter estimates (on the 'real probability scale') by clicking on the fourth button (from the left) on the menu, or by selecting '**Output | Specific Model Output | Parameter Estimates | Real**'.

This spawns a Windows Notepad window to open up, containing the estimates (shown below).



mrk4718z.tmp - Notepad

File Edit Format View Help

European Dipper Data - Males Only

Real Function Parameters of {phi(t)p(t)}

Parameter	Estimate	Standard Error	95% Confidence Interval Lower	Upper
1:Phi	0.6969698	0.2049829	0.2555234	0.9390713
2:Phi	0.4230769	0.0968907	0.2519587	0.6148826
3:Phi	0.5052879	0.0874934	0.3396458	0.6697773
4:Phi	0.6094017	0.0838412	0.4389147	0.7567907
5:Phi	0.5708181	0.0776952	0.4166817	0.7123428
6:Phi	0.7637583	514.12938	0.4489907E-10	1.0000000
7:p	0.7173912	0.2238535	0.2257359	0.9567135
8:p	1.0000000	0.5048141E-07	0.9999999	1.0000001
9:p	0.9093023	0.0855744	0.5674379	0.9871171
10:p	0.9274193	0.0693437	0.6291492	0.9897162
11:p	0.9358289	0.0616791	0.6607895	0.9909235
12:p	0.7637669	514.13517	0.4490122E-10	1.0000000

The **MARK** output consists of 5 columns: the first column is the parameter index number ($1 \rightarrow 12$), followed by the parameter estimate, the standard error of the parameter, and the lower and upper 95% confidence limits for the estimates, respectively. The parameter indexing relates to the indexing used in the PIMs we saw earlier.

3.4.1. MARK, PIMs, and parameter indexing

Let's stop a moment for a quick introduction to the indexing scheme that **MARK** uses. Consider the following figure:

$$\mathbf{1} \overset{\varphi_1}{\longrightarrow} \underset{p_2}{\mathbf{2}} \overset{\varphi_2}{\longrightarrow} \underset{p_3}{\mathbf{3}} \overset{\varphi_3}{\longrightarrow} \underset{p_4}{\mathbf{4}} \overset{\varphi_4}{\longrightarrow} \underset{p_5}{\mathbf{5}} \overset{\varphi_5}{\longrightarrow} \underset{p_6}{\mathbf{6}} \overset{\varphi_6}{\longrightarrow} \underset{p_7}{\mathbf{7}}$$

In our analysis of the male Dipper data, recall that we have 7 occasions - the initial marking occasion, and 6 subsequent recapture (or resighting) occasions. The φ_i values represent the survival probabilities between successive occasions, while the p_i values represent the recapture probabilities at the ith occasion. For details, see Lebreton *et al*. (1992).

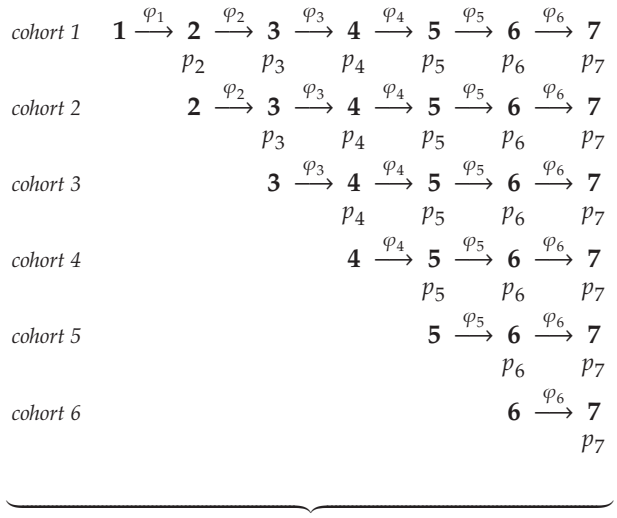
What **MARK** does is to substitute a numerical indexing scheme for the individual φ_i and p_i values, respectively. For example, consider the indexing for the survival parameters φ_i - there are 6 values, φ_1 through φ_6. How does this connect to the 'survival PIM'?

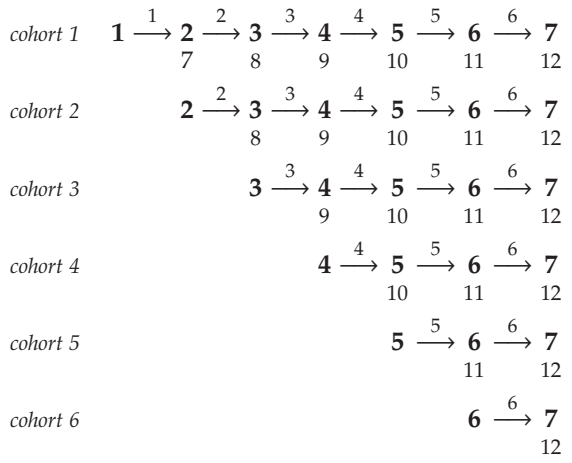
$$\mathbf{1} \overset{\varphi_1}{\longrightarrow} \underset{p_2}{\mathbf{2}} \overset{\varphi_2}{\longrightarrow} \underset{p_3}{\mathbf{3}} \overset{\varphi_3}{\longrightarrow} \underset{p_4}{\mathbf{4}} \overset{\varphi_4}{\longrightarrow} \underset{p_5}{\mathbf{5}} \overset{\varphi_5}{\longrightarrow} \underset{p_6}{\mathbf{6}} \overset{\varphi_6}{\longrightarrow} \underset{p_7}{\mathbf{7}}$$

$$\Downarrow$$

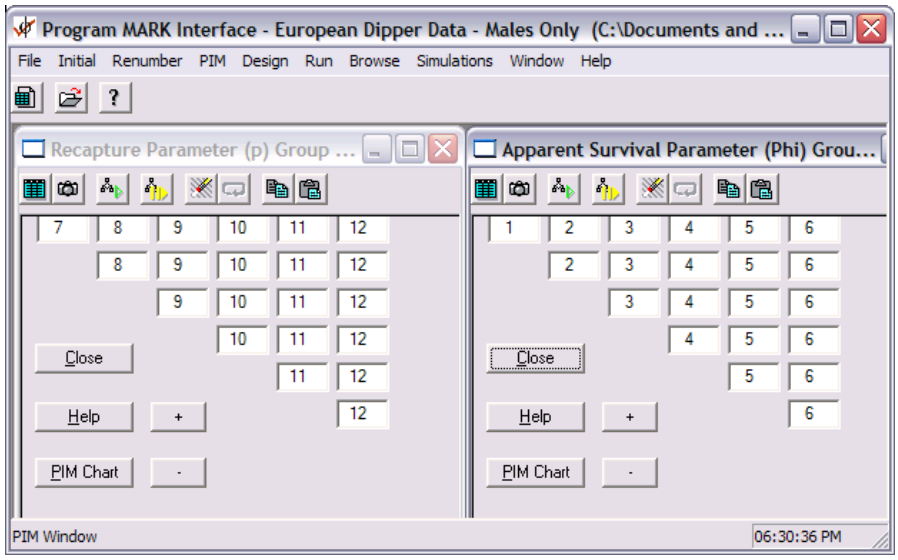
$$\mathbf{1} \overset{1}{\longrightarrow} \underset{7}{\mathbf{2}} \overset{2}{\longrightarrow} \underset{8}{\mathbf{3}} \overset{3}{\longrightarrow} \underset{9}{\mathbf{4}} \overset{4}{\longrightarrow} \underset{10}{\mathbf{5}} \overset{5}{\longrightarrow} \underset{11}{\mathbf{6}} \overset{6}{\longrightarrow} \underset{12}{7}$$

What about individuals captured for the first time and marked at the second occasion? Technically, we would refer to such individuals as being in the second release *cohort* (a cohort is simply a group of individuals that bear something in common - in this case, individuals captured, marked and released on the second occasion would comprise the second cohort). This is similar for individuals captured, marked and released on the third occasion, and so forth. All we need to do to account for these different release cohorts is to start the 'indexing' at the appropriate occasion for each cohort – this is shown schematically, below:





Now, have another look at the PIMs for survival and recapture (at the top of the next page). The numbers 1 to 6 in the PIM correspond to the survival values φ_1 to φ_6, respectively. For the recapture PIM, the numbers 7 to 12 correspond to the recapture probabilities p_2 to p_7, respectively. Notice that **MARK** indexes the survival parameters first (1 to 6), followed in numerical sequence by the index values for the recaptures (7 to 12). In other words, the indexing that **MARK** uses does not correspond to the number of the particular interval or occasion involved. For example, the recapture index 8 corresponds to the recapture probability at occasion 3 (i.e., p_3). Although it can be a little confusing at first, with a bit of work you should be able to see the basic connection between the 'true' parameter structure of the model, and the way in which **MARK** indexes it, both internally (which becomes very important later on as we examine more complex models), and in the output file.



It is very important that you grasp this connection, so take some time now to ensure that you do...

We'll be using PIMs frequently throughout the rest of the book, so not to worry if you don't immediately see the connection - you'll get lots of practice. But, make sure you spend some time trying to grasp the connection now.

So, getting back to the output:

mrk4718z.tmp - Notepad

File Edit Format View Help

European Dipper Data - Males Only

Real Function Parameters of {phi(t)p(t)}

Parameter	Estimate	Standard Error	95% Confidence Interval Lower	Upper
1:Phi	0.6969698	0.2049829	0.2555234	0.9390713
2:Phi	0.4230769	0.0968907	0.2519587	0.6148826
3:Phi	0.5052879	0.0874934	0.3396458	0.6697773
4:Phi	0.6094017	0.0838412	0.4389147	0.7567907
5:Phi	0.5708181	0.0776952	0.4166817	0.7123428
6:Phi	0.7637583	514.12938	0.4489907E-10	1.0000000
7:p	0.7173912	0.2238535	0.2257359	0.9567135
8:p	1.0000000	0.5048141E-07	0.9999999	1.0000001
9:p	0.9093023	0.0855744	0.5674379	0.9871171
10:p	0.9274193	0.0693437	0.6291492	0.9897162
11:p	0.9358289	0.0616791	0.6607895	0.9909235
12:p	0.7637669	514.13517	0.4490122E-10	1.0000000

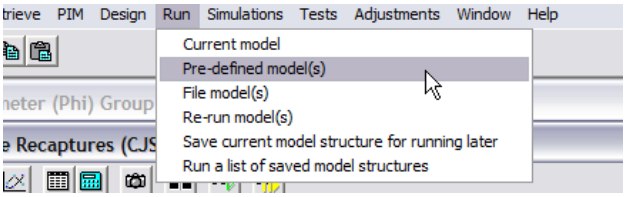
Note that our survival estimates correspond to parameters 1 to 6, while 7 to 12 correspond to the recapture probabilities (corresponding to the figures on the preceding page). But, if you look carefully, you'll notice that the estimates for index 6 (the survival probability from occasion 6 to occasion 7) and index 12 (the recapture probability at occasion 7) are identical (0.76377). As discussed in detail in Lebreton *et al*. (1992), and elsewhere in this book, this reflects the fact that, for this model, the survival and recapture rates for the last interval are not individually identifiable. The value of 0.76377 is actually

the square-root of the estimated product $\varphi_6 p_7$ (this is denoted as β_7 in Lebreton *et al*. 1992). Thus, **MARK** has estimated 11 parameters - 5 survival probabilities (φ_1 to φ_5), 5 recapture probabilities (p_2 to p_6), and one related to the product $\varphi_6 p_7$. You may also notice that the SE and 95% CI for one of the estimates ($\hat{p}_3$) is clearly 'suspect' - we'll deal much more with 'problems with the estimates' later.

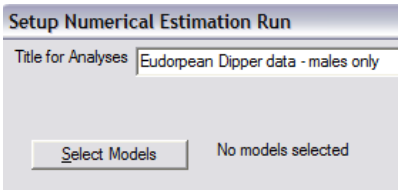
At this point, you can do one of several things. You can print your estimates, you can plot them, or you can examine aspects of the estimation procedure, and look at the degree to which any given capture history in your `.INP` file affects your estimates. We'll defer all the options for the moment, and simply close the notepad window with the estimates, and look back at the main browser window itself.

Much of what **MARK** shows you in the browser is important only in the context of comparing the fit of one model with that of another. In order to demonstrate this, we'll continue our exploration of the Dipper data set, by running 3 additional models: $\{\varphi_t, p\}$, $\{\varphi, p_t\}$ and $\{\varphi, p\}$, corresponding to time-varying survival, constant recapture, constant survival, time-varying recapture, and constant survival and recapture, respectively. There are **many** more models we could try to fit, but for the purposes of this exercise (which is intended to give you a more complete sense of how the results browser works), we'll run these three. There are a number of ways you could specify these models in **MARK**. For the moment, we'll use a 'short-cut' method which is convenient for some standard models. The short-cut makes use of some 'built-in' models in **MARK**. **Note**: you will be **strongly** discouraged from ever using this short-cut again, but for the moment, it's convenient.

If you pull down the '**Run**' menu, you will see that there are several menu options, beside the 'run the current model' we used earlier. The two options of interest at this point are: '**pre-defined**' models and '**file**' models. For the moment, we're interested only in the pre-defined models.

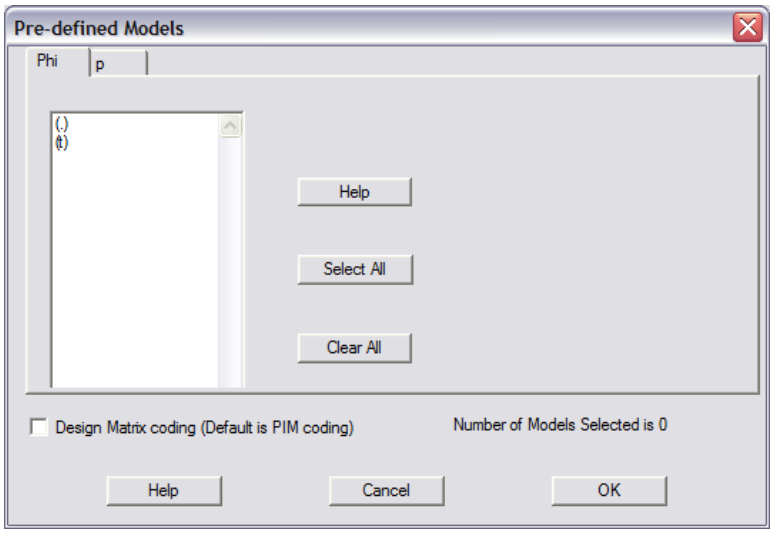


Once you have selected the '**pre-defined models**' option from the '**Run**' menu, you will be dumped directly back into the '**Run**' window. However, there is one important, but easily overlooked change to the window. Where in the first instance there was a button to fix parameter estimates, note that now this button has been replaced with a '**Select models**' button. Everything else is the same, so you have to be observant. The other difference is that the model title box has been eliminated. In a moment you'll see why. Click on the '**Select models**' button.



Once you've clicked on the appropriate button, you'll be presented with a tabbed-window (shown at the top of the next page), where each tab represents one of the parameters in the models you're working with (in this case, φ and p). All you need to do is (i) select the parameter by clicking the appropriate tab, and (ii) select the model structure(s) you want for each parameter in turn. For example, the following

shows the φ parameter – we've selected only the time-invariant 'dot' model {.}. (Don't worry about the 'design matrix' option for the moment - much more on that in Chapter 6). Selecting both t and . for both parameters would yield 4 final models: $\{\varphi_t p_t\}$, $\{\varphi_t p_.\}$, $\{\varphi_. p_t\}$, and $\{\varphi_. p_.\}$. For the moment, we're only interested in the last 3 (since we've already built model $\{\varphi_t p_t\}$).

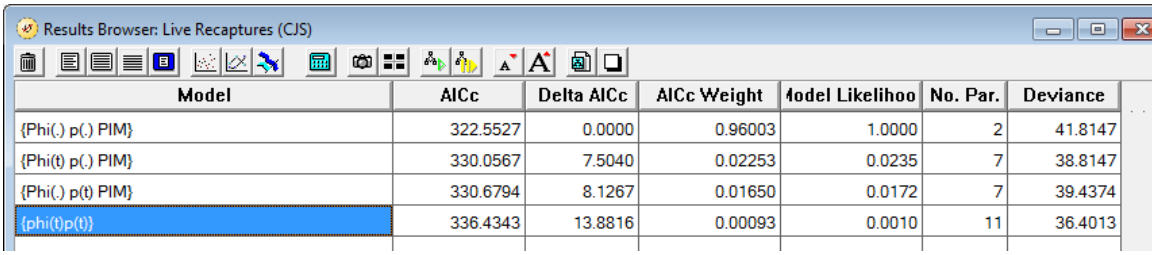


So, simply select the {.} option for the φ parameter, as shown above. Note that at this point, **MARK** tells you (in the lower-right hand corner of the window) that the number of models selected to run is 0 – this is because we haven't yet defined the structure for the other parameter p yet. Click the tab for the p parameter, and select both the {t} and {.} models. Now, **MARK** reports that you've selected 2 models: $\{\varphi_. p_.\}$ and $\{\varphi_. p_t\}$. Click the '**OK**' button for the 2 models we've just selected. This brings you back to the '**Set Up Numerical Estimation Run**' window again. Click the '**Run**' button, which will run these 2 models, and automatically add the results to the browser. However, we also want to run model $\{\varphi_t p_.\}$. Simply go through this process again (i.e., running a pre-defined model), except this time, select {t} for the φ parameter, and {.} for the p parameter. Click '**OK**', and run this model, adding the results to the browser.

Note that when running pre-defined models, you still do not have the option of fixing parameters, and there is still no input box to specify the model title. The reason? Simple – **MARK** figures that if you're using the pre-defined models, you're willing to accept the parameter structure and model names that **MARK** uses for defaults. Of course, **MARK** still allows you to choose among the various link functions, variance estimation routines, and other options normally associated with the run window. If you're satisfied with what is set for these other options, then simply click the '**OK to run**' button in the lower right-hand corner of the run window.

Running pre-defined models in **MARK** is somewhat different from the way they run normally (i.e., the way our initial model ran). First, running pre-defined windows does not cause **MARK** to spawn a command window showing the progress of the numerical estimation. Second, **MARK** does not ask you after each model is finished whether or not you want to add the results to the results browser - **MARK** assumes you do, and sends the results to the browser automatically.

Once the processing of the models is complete, you will note that they have been added to the results browser:



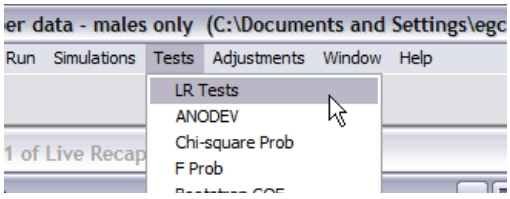
Model	AICc	Delta AICc	AICc Weight	Model Likelihood	No. Par.	Deviance
{Phi(.) p(.) PIM}	322.5527	0.0000	0.96003	1.0000	2	41.8147
{Phi(t) p(.) PIM}	330.0567	7.5040	0.02253	0.0235	7	38.8147
{Phi(.) p(t) PIM}	330.6794	8.1267	0.01650	0.0172	7	39.4374
{phi(t)p(t)}	336.4343	13.8816	0.00093	0.0010	11	36.4013

However, note that the results are not listed in the order in which the models were processed. In fact, if you'd been watching the results browser while **MARK** was processing the list of models, you might have noticed that the ordering of the models in the list changed with the completion of each model. In fact, **MARK** is ordering (or sorting) the results based on some sort of criterion - in this case, in ascending sequence starting with the model with the lowest AIC_c value (for the Dipper example, this corresponds to model '`Phi(.)p(.)`' – constant survival and constant recapture). The sorting criterion can be controlled using the '**`Order`**' menu.

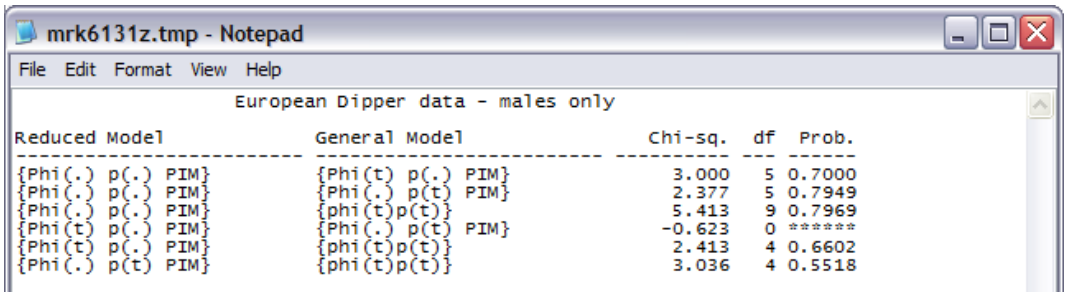
Although we will talk a **lot** more about AIC_c in subsequent chapters, for the moment, take it on faith that the AIC_c is a good, well-justified criterion for selecting the most parsimonious model (i.e., the model which best explains the variation in the data while using the fewest parameters). In a very loose sense, we might state that the model with the lowest AIC_c is the 'best' model (although clearly, what is 'best' or 'worst' depends upon the context). The results browser shows you the AIC for each model, as well as the arithmetic difference between each model and the top model (i.e., the one with the lowest AIC_c value). For example, model '`Phi(t)p(.)`' has an AIC_c value of 330.06, which is 7.50 units larger than the AIC_c for the most parsimonious model (model '`Phi(.)p(.)`', which has an AIC_c value of 322.55).

The right-most column (by default) is the model deviance. In simple terms, the lower the deviance, the better the model fits. The technical details of the estimation of the likelihood and deviances are given in Lebreton *et al*. (1992). We'll talk more about the deviance (and related statistics) later.

Perhaps more notably, the difference in deviance between 'nested models' (models in which one model differs from another by the elimination of one or more model terms) is distributed as a χ^2 statistic with the degrees of freedom given as the difference in the number of estimable parameters between the two models. This forms the basis of the likelihood ratio test (LRT). In fact, **MARK** provides a variety of 'statistical tests' for comparing among models, including the LRT. To perform a LRT on the models in the results browser, simply pull down the '**`Tests`**' menu, and select '**`LR tests`**' (as shown at the top of the next page). (*Note*: the relationship between AIC, model selection, and 'classical' statistical tests like the LRT will be presented in more detail in subsequent chapters.)



MARK will then present you with a window allowing you to select the models you want to compare using LRT. For now, simply '**Select all**', and then click the '**OK**' button. You will be shown the results of the LRT tests in a Notepad window.



Now, what do we note about the results? Most importantly, we see that not all paired-comparisons among models are possible - the comparison between model $\{\varphi_t,p\}$ and model $\{\varphi,p_t\}$ is not calculated. Why? If you recall from the preceding page, LRT may be applied only to 'nested' models. We'll talk more about 'nesting' in the next chapter, but for now, accept that these 2 models are not nested. As such, we cannot use an LRT to compare the fit of the 2 models. In one sense (although perhaps not the most appropriate one), this is one of the advantages of using the AIC to compare models - it works regardless of whether or not the models are nested. [However, as we will discover in later chapters, there may be far more important reasons to use AIC as an omnibus (general) model selection tool.]

For the moment, concentrate on the comparison of model $\{\varphi_{.}p_{.}\}$ with model $\{\varphi_t p_{.}\}$, the first 2 models noted in the notepad window (above). We note that the difference in the model deviances is 3, with a difference in the number of parameters of 5. Based on a χ^2 distribution, this difference is not significant at the nominal $\alpha = 0.05$ level ($P = 0.700$). In other words, both models fit the data equally well (we cannot differentiate between them statistically). The model with more parameters fits the data better in terms of the model deviance, but not so much so as to compensate for the fact that it takes more parameters to achieve this better fit. Thus, we would conclude that model $\{\varphi_{.}p_{.}\}$ is the more parsimonious model, which is consistent with the results using the AIC_c.

Of course, at this point you can browse the estimates, plot them, examine residual plots - simply by selecting the model you're interested in (by clicking on it in the results browser). You can also re-run any particular model, using (for example) a different link function, simply by selecting the model from the list, retrieving the model (the '**Retrieve**' menu), and then re-running the model (specifying the new link function, of course). **MARK** will process the data, and then ask you if you want to add the new results to the browser. It's that easy.

Now, for the final steps. Once you're done working with this problem, all you need to do is exit **MARK**. All the model results will be stored away in a `DBF` file (in this case, called `ED_MALES.DBF`). Then, if you want to continue work on this analysis, all you'll need to do is start **MARK**, and then '**Open**' the `ed_males.dbf` file. That's it!

3.5. Summary

Congratulations! You've just finished your first analysis using program **MARK**. Of course, the fact that there are many hundreds of pages left in this book should tell you that there is a lot more left to be covered. But, you've at least gotten your feet wet, and run through a 'typical' **MARK** analysis once - this is an important first step. If you don't feel comfortable with what we've done so far go back through the chapter - slowly. Many of the basic mechanics and presented in this chapter (in particular, the relationship between model 'structure' and the PIMs) will be used repeatedly throughout the rest of the book, so it is important to feel comfortable with them before proceeding much further.

CHAPTER 4

Building & comparing models

In this chapter, we introduce several important concepts.* First, we introduce the basic concepts and 'mechanics' for building models in **MARK**. We demonstrate models for both single and multiple groups of marked individuals. Second, we introduce some of the concepts behind the important questions of 'model selection' and 'multi-model inference'. *How* to build models in **MARK** is 'mechanics' - *why* we build certain models, and what we do with them, is 'science'. Both are *critical* concepts to master in order to use **MARK** effectively, and are *fundamental* to understanding everything else in this book, so take your time.

We'll begin by re-visiting the male Dipper data we introduced in the last chapter. We will compare 2 different subsets of models: models where either survival or recapture (or both) varies with time, or models where either survival or recapture (or both) are constant with respect to time. The models are presented in the following table, using the notation suggested in Lebreton *et al*. (1992).

model	*explanation*
$\{\varphi_t p_t\}$	both survival and encounter probability time dependent
$\{\varphi_. p_t\}$	survival constant over time, encounter probability time dependent
$\{\varphi_t p_.\}$	survival time dependent, encounter probability constant over time
$\{\varphi_. p_.\}$	both survival and encounter probabilities constant over time

In the following, we will go through the steps in fitting each of these 4 models to the data. In fact, these models are the same ones we fit in Chapter 3. So why do them again? In Chapter 3, our intent was to give you a (very) gentle run-through of running **MARK**, using some of the standard options. In this chapter, the aim is to introduce you to the mechanics of model building, from the ground up. We will not rely on 'built-in' or 'pre-defined' models in this chapter (in fact, you're not likely to ever use them again). Since you already saw the 'basics' of getting **MARK** up and running in Chapter 3, we'll omit some of the more detailed explanations for each step in this chapter.

However, we must emphasize that before you actually use **MARK** (or any other program) to compare different models, you need to first confirm that your 'starting model' (generally, the most parameterized or most general model) adequately fits the data. In other words, you **must** conduct a goodness-of-fit (GOF) test for your 'starting model'. GOF testing is discussed in detail in Chapter 5, and periodically throughout the remainder of this book. For convenience, we'll assume in this chapter that the 'starting

* Very important...

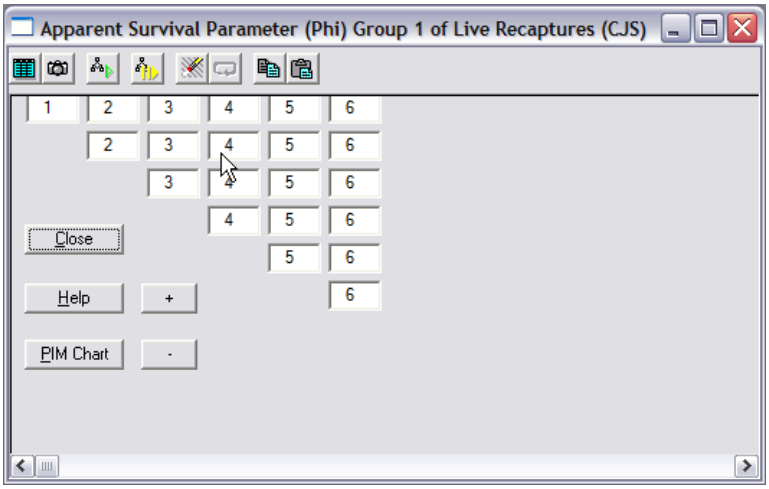
model' does adequately fit the data.

4.1. Building models - parameter indexing & model structures

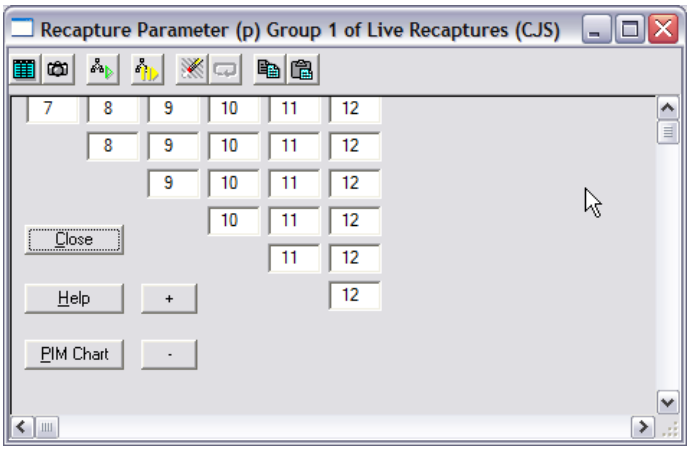
As in Chapter 3, start **MARK** by double-clicking the **MARK** icon. We're going to use the same data set we analyzed in Chapter 3 (`ED_MALES.INP`). At this point, we can do one of 2 things: (1) we can start a completely new **MARK** session (i.e., create a completely new `*.DBF` file), or (2) we can re-open the `*.DBF` file we created in Chapter 3, and append new model results to it. Since you already saw in Chapter 3 how to start a 'new' project, we'll focus here on the second possibility - appending new model results to the existing `*.DBF` file.

This is very easy to do - from the opening **MARK** 'splash screen', select '**`Open`**' from the '**`File`**' menu, and find the `ED_MALES.DBF` file you created in Chapter 3 (remember that **MARK** uses the prefix of the `*.INP` file - the file containing the encounter histories - as the prefix for the `*.DBF` file. Thus, analysis of `ED_MALES.INP` leads to the creation of `ED_MALES.DBF`). Once you've found the `ED_MALES.DBF` file, simply double-click the file to access it. Once you've double-clicked the file, the **MARK** 'splash screen' will disappear, and you'll be presented with the main **MARK** window, and the results browser. In the results browser, you'll see the models you already fit to these data in the last chapter (there should be 4 models), and their respective AIC and deviance values.

In this chapter, we want to show you how to build these models from scratch. As such, there is no point in starting with all the results already in the browser! So, take a deep breath and delete all the models currently in the browser! To do this, simply highlight each of the models in turn, and click the trash-can icon in the browser toolbar. Next, bring up the *Parameter Index Matrices* (PIMs), which (as you may recall from Chapter 3), are fundamental to determining the structure of the model you are going to fit to the data. So, the first step is to open the PIMs for both the survival and recapture parameters. To do this, simply pull down the '**`PIM`**' menu, and select '**`Open Parameter Index Matrix`**'. This will present you with a dialog box containing two elements: '**`Apparent Survival Parameter (Phi) Group 1`**', and '**`Recapture Parameter (p) Group 1`**'. You want to select both of them. You can do this either by clicking on both parameters, or by simply clicking on the '**`Select All`**' button on the right-hand side of the dialog box. Once you've selected both PIMs, simply click the '**`OK`**' button in the bottom right-hand corner. This will cause the PIMs for survival and recapture to be added to the **MARK** window. Here's what they look like, for the survival parameters



and the recapture parameters, respectively.



We're going to talk a **lot** more about PIMs, and why they look like they do, later on in this (and subsequent) chapters. For now, the only thing you need to know is that these PIMs reflect the currently active model. Since you deleted all the models in the browser, **MARK** reverts to the default model - which is **always** the fully time-dependent model. For mark-recapture data, this means the fully time-dependent CJS model.

OK, so now you want to fit a model. While there are some 'built-in' models in **MARK**, we'll concentrate at this stage on using **MARK** to manually build the various models we want to fit to our data. Once you've mastered this manual, more general approach, you can then proceed to using 'short-cuts' (such as built-in models). Using short-cuts before you know the 'general way' is likely to lead to one thing - you getting lost!

Looking back at the table on the first page of this chapter, we see that we want to fit 4 models to the data, $\{\varphi_t p_t\}$, $\{\varphi_t p_.\}$, $\{\varphi_. p_t\}$ and $\{\varphi_. p_.\}$. A quick reminder about model syntax - the presence of a 't' subscript means that the model is structured such that estimates for a given parameter are time-specific; in other words, that the estimates may differ over time. The absence of the 't' subscript (or, the presence of a 'dot') means the model will assume that the parameter is fixed through time (the use of the 'dot' subscript leads to such models usually being referred to as 'dot models' – naturally).

Let's consider model $\{\varphi_t p_t\}$ first. In this model, we assume that both survival (φ) and recapture (p) can vary through time. How do we translate this into **MARK**? Pretty easy, in fact. First, recall that in this data set, we have 7 total occasions: the first occasion is the initial marking (or release) occasion, followed by 6 subsequent recapture occasions. Now, typically, in each of these subsequent recapture occasions 2 different things can occur. Obviously, we can recapture some of the individuals previously marked. However, part of the sample captured on a given occasion is unmarked. What the investigator does with these individuals differs from protocol to protocol. Commonly, all unmarked individuals are given a unique mark, and released. As such, on a given recapture occasion, 2 types of individuals are handled and released: those individuals which have been previously marked, and those which are newly marked. Whether or not the fate of these two 'types' of individuals is the same is something we can test (we will explore this in a later chapter). In some studies, particularly in some fisheries and insect investigations, individuals are only marked at the initial release (sometimes known as a 'batch mark'). There are no newly marked individuals added to the sample on any subsequent occasions. The distinctions between these two types of mark-release schemes are important to understanding the structure of the parameter matrices **MARK** uses.

Consider our first model, the CJS model $\{\varphi_t p_t\}$ with full time-dependence in both survival and recapture probabilities. Let's assume there are no age effects (say, for example, all individuals are marked as adults – we deal with 'age' in a later chapter). In Chapter 3, we represented the parameter structure of this model as shown below:

$$\mathbf{1} \xrightarrow{\varphi_1} \underset{p_2}{\mathbf{2}} \xrightarrow{\varphi_2} \underset{p_3}{\mathbf{3}} \xrightarrow{\varphi_3} \underset{p_4}{\mathbf{4}} \xrightarrow{\varphi_4} \underset{p_5}{\mathbf{5}} \xrightarrow{\varphi_5} \underset{p_6}{\mathbf{6}} \xrightarrow{\varphi_6} \underset{p_7}{\mathbf{7}}$$

In fact, this representation is incomplete, since it does not record or index the fates of individuals newly marked and released at each occasion. These are referred to as 'cohorts' - groups of animals marked and released on a particular occasion.

We can do this easily by adding successive rows to our model structure, each row representing the individuals newly marked on each occasion. Since the occasions obviously occur sequentially, then each row will be indented from the one above it by one occasion. This is shown below:

$$\begin{array}{ll}
\textit{cohort 1} & \mathbf{1} \xrightarrow{\varphi_1} \underset{p_2}{\mathbf{2}} \xrightarrow{\varphi_2} \underset{p_3}{\mathbf{3}} \xrightarrow{\varphi_3} \underset{p_4}{\mathbf{4}} \xrightarrow{\varphi_4} \underset{p_5}{\mathbf{5}} \xrightarrow{\varphi_5} \underset{p_6}{\mathbf{6}} \xrightarrow{\varphi_6} \underset{p_7}{\mathbf{7}} \\
\textit{cohort 2} & \mathbf{2} \xrightarrow{\varphi_2} \underset{p_3}{\mathbf{3}} \xrightarrow{\varphi_3} \underset{p_4}{\mathbf{4}} \xrightarrow{\varphi_4} \underset{p_5}{\mathbf{5}} \xrightarrow{\varphi_5} \underset{p_6}{\mathbf{6}} \xrightarrow{\varphi_6} \underset{p_7}{\mathbf{7}} \\
\textit{cohort 3} & \mathbf{3} \xrightarrow{\varphi_3} \underset{p_4}{\mathbf{4}} \xrightarrow{\varphi_4} \underset{p_5}{\mathbf{5}} \xrightarrow{\varphi_5} \underset{p_6}{\mathbf{6}} \xrightarrow{\varphi_6} \underset{p_7}{\mathbf{7}} \\
\textit{cohort 4} & \mathbf{4} \xrightarrow{\varphi_4} \underset{p_5}{\mathbf{5}} \xrightarrow{\varphi_5} \underset{p_6}{\mathbf{6}} \xrightarrow{\varphi_6} \underset{p_7}{\mathbf{7}} \\
\textit{cohort 5} & \mathbf{5} \xrightarrow{\varphi_5} \underset{p_6}{\mathbf{6}} \xrightarrow{\varphi_6} \underset{p_7}{\mathbf{7}} \\
\textit{cohort 6} & \mathbf{6} \xrightarrow{\varphi_6} \underset{p_7}{\mathbf{7}}
\end{array}$$

Notice that the occasions are numbered from left to right, starting with occasion 1. Survival probability is the probability of surviving between successive occasions (i.e., between columns). Each release cohort is listed in the left-hand column.

For example, some individuals are captured and marked on occasion 1, released, and potentially can survive to occasion 2. Some of these surviving individuals may survive to occasion 3, and so on. At occasion 2, some of the captured sample are unmarked. These unmarked individuals are newly marked and released at occasion 2. These animals comprise the second release cohort. At occasion 3, we take a sample from the population. Some of the sample might consist of individuals marked in the first cohort (which survived to occasion 3), some would consist of individuals marked in the second cohort (which survived to occasion 3), while the remainder would be unmarked. These unmarked individuals are newly marked, and released at occasion 3. These newly marked and released individuals comprise the third release cohort. And so on.

If we rewrite cohort structure, showing only the occasion numbers, we get the structure shown at the top of the next page.

cohort 1	**1**	**2**	**3**	**4**	**5**	**6**	**7**
cohort 2		**2**	**3**	**4**	**5**	**6**	**7**
cohort 3			**3**	**4**	**5**	**6**	**7**
cohort 4				**4**	**5**	**6**	**7**
cohort 5					**5**	**6**	**7**
cohort 6						**6**	**7**

The first question that needs to be addressed is: does survival vary as a function of which cohort an individual belongs to, does it vary with time, or both? This will determine the indexing of the survival and recapture parameters. For example, assume that cohort does not affect survival, but that survival varies over time. In this case, survival can vary among intervals (i.e., among columns), but over a given interval (i.e., within a column), survival is the same over all cohorts (i.e., over all rows). Again, consider the following cohort matrix - but showing only the survival parameters:

cohort 1	**1**	φ_1	**2**	φ_2	**3**	φ_3	**4**	φ_4	**5**	φ_5	**6**	φ_6	**7**
cohort 2			**2**	φ_2	**3**	φ_3	**4**	φ_4	**5**	φ_5	**6**	φ_6	**7**
cohort 3					**3**	φ_3	**4**	φ_4	**5**	φ_5	**6**	φ_6	**7**
cohort 4							**4**	φ_4	**5**	φ_5	**6**	φ_6	**7**
cohort 5									**5**	φ_5	**6**	φ_6	**7**
cohort 6											**6**	φ_6	**7**

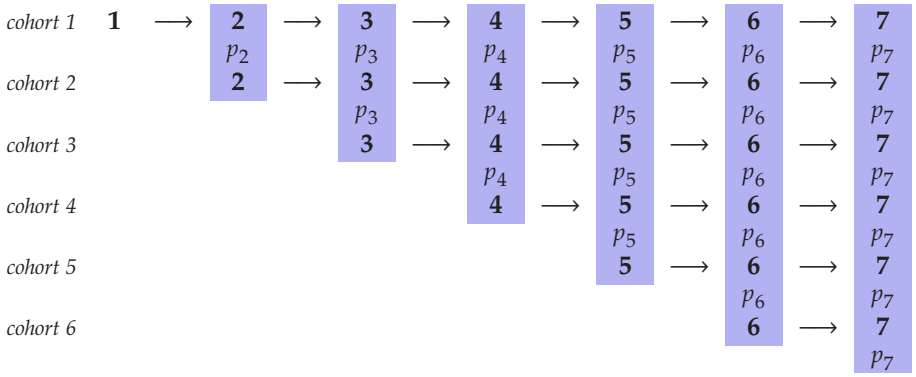
The shaded columns indicate that survival is constant over cohorts, but the changing subscripts in φ_i indicate that survival may change over time. This is essentially Table 7A in Lebreton *et al*. (1992). What **MARK** does to generate the parameter or model structure matrix is to reproduce the structure and dimensions of this figure, after first replacing the φ_i values with a simple numerical indexing scheme, such that φ_1 is replaced by the number 1, φ_2 is replaced by the number 2, and so forth. Thus, the preceding figure (above) is represented by a triangular matrix of the numbers 1 to 6 (for the 6 survival probabilities):

1	2	3	4	5	6
	2	3	4	5	6
		3	4	5	6
			4	5	6
				5	6
					6

This 'triangular matrix' (the PIM) represents the way that **MARK** 'stores' the model structure corresponding to time variation in survival, but no cohort effect (Fig. 7A in Lebreton *et al*. 1992). Notice that the dimension of this matrix is (6 rows by 6 columns), rather than (7 columns by 7 rows). This is because there are 7 capture occasions, but only 6 survival intervals (and, correspondingly, 6 recapture occasions). This representation is the basis of the PIMs which you see on your screen (it will also be printed in the output file). Perhaps most importantly, though, this format is the way **MARK** keeps track of model

structure and parameter indexing. It is essential that you understand the relationships presented in the preceding figures. A few more examples will help make them clearer.

Let's consider the recapture probability. If recapture probability is also time-specific, what do you think the model structure would look like? If you've read **and** understood the preceding, you should be able to make a reasonable guess. Again, remember that we have 7 occasions - the initial marking event (occasion 1), and 6 recapture occasions. With time-dependence, and assuming no differences among cohorts, the model structure for recaptures would be:



Now, what are the corresponding index values for the recapture probabilities? As with survival, there are 6 parameters, p_2 to p_7 (corresponding to recapture on the second through seventh occasion, respectively). With survival probabilities, we simply looked at the subscripts of the parameters, and built the PIM. However, things are not quite so simple here (although as you'll see, they're not very hard). All you need to know is that the recapture parameter index values start with the first value after the survival values. Hmmm...let's try that another way. For survival, we saw there were 6 parameters, so our survival PIM looked like

1	2	3	4	5	6
	2	3	4	5	6
		3	4	5	6
			4	5	6
				5	6
					6

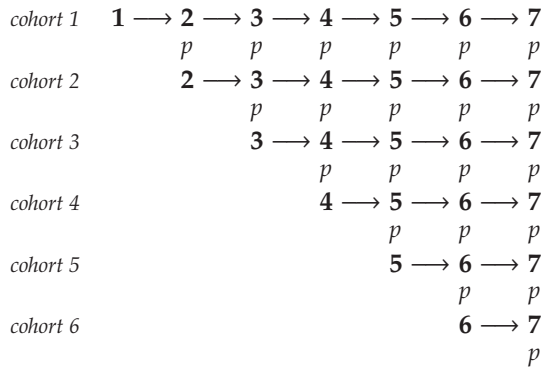
The last index value is the number '6' (corresponding to φ_6, the apparent survival probability between occasion 6 and occasion 7). To build the recapture PIM, *we start with the first value after the largest value in the survival PIM*. Since '6' is the largest value in the survival PIM, then the first index value used in the recapture PIM will be the number '7'. Now, we build the rest of the PIM. What does it look like?

If you think about it for a moment, you'll realize that the recapture PIM looks like:

7	8	9	10	11	12
	8	9	10	11	12
		9	10	11	12
			10	11	12
				11	12
					12

Do these look familiar? They might - look at the PIMs **MARK** has generated on the screen. In fact, we're now ready to 'run the CJS model' fully time-dependent model. We covered this step in Chapter 3, but let's go through it again (repetition is a good teacher). In fact, there are a couple of ways you can proceed. You can either (i) pull down the '**Run**' menu and '**Run the current model**' (the model defined by the PIMs is always the current model), or (ii) click the '**Run**' icon on the toolbar of either of the PIMs. This will bring up the '**Setup**' window for the numerical estimation, which you saw for the first time in Chapter 3. All you need to do is fill in a name for the model (we'll use `Phi(t)p(t)` for this model), and click the '**OK to run button**' (lower right-hand corner). Again, as you saw in Chapter 3, **MARK** will ask you about the 'identity matrix', and then spawn a numerical estimation window. Once it's finished, simply add these results to the results browser.

Now, let's consider model $\{\varphi_t p_\cdot\}$ - time-dependent survival, but constant recapture probability. What would the PIMs for this model look like? Well, clearly, the survival PIM would be identical to what we already have, so no need to do anything there. What about the recapture PIM? Well, in this case, we have constant recapture probability. What does the parameter structure look like? Look at the following figure:



Note that there are no subscripts for the recapture parameters - this reflects the fact that for this model, we're setting the recapture probability to be constant, both among occasions, and over cohorts.

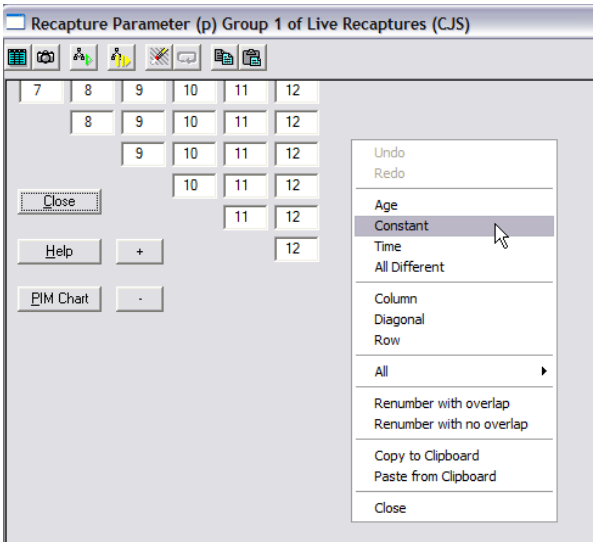
What would the PIM look like for recapture probability? Well, recall that the largest index value for the survival PIM is the number '6', so the first index value in the recapture PIM is the number '7'. And, since the recapture probability is constant for this model, then the entire PIM will consist of the number '7' (as shown at the top of the next page).

```
7  7  7  7  7  7
   7  7  7  7  7
      7  7  7  7
         7  7  7
            7  7
               7
```

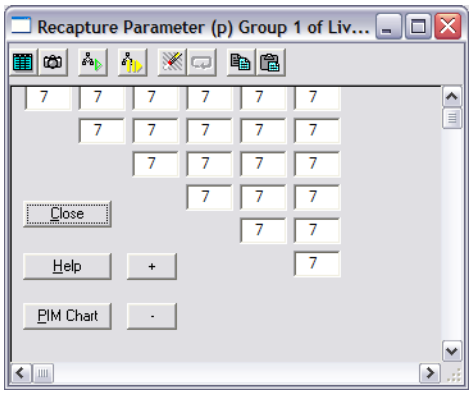
Now, how do we modify the PIMs in **MARK** to reflect this structure? As you'll discover, **MARK** gives you many different ways to accomplish the same thing. Modifying PIMs is no exception. The most obvious (and pretty well fool-proof) way to modify the PIM is to edit the PIM directly, changing each cell in the PIM, one at a time, to the desired value. For small PIMs, or for some esoteric model structures we'll discuss in later chapters, this is not a bad thing to try. However, let's use one of the built-in time-savers in **MARK** to do most of the work for us.

Remember, all we want to do here is modify the recapture PIM. To do this, make that PIM 'active' by clicking in the first 'cell' (upper left corner of the PIM). You can in fact make a window active by clicking on it anywhere (it doesn't matter where - just remember not to click the '**X**' in the upper right-hand corner, since this will close the window!), but as we'll see, there are advantages in clicking in a specific cell in the PIM. When you've successfully selected a cell, you should see a vertical cursor in that cell.

Once you've done this, you can do one of a couple of things. You can pull down the '**Initial**' menu on the main **MARK** parent toolbar. When you do this, you'll see a number of options - each of them controlling the value (if you want, the initial value) of some aspect of the active window (in this case, the recapture PIM). Since we want to have a constant recapture probability, you might guess the '**Constant**' option on the '**Initial**' menu would be the right one. You'd be correct. Alternatively, you can right-click with the mouse anywhere in the recapture PIM window - this will generate the same menu as you would get if you pull down the '**Initial**' menu. Use whichever approach you prefer:



Once you've done this, you will see that all the values in the recapture PIM are changed to 7.



Believe it our not, you're now ready to run this model (model $\{\varphi_t p_.\}$). Simply go ahead and click the '**Run**' icon in the toolbar of either PIM. For a model name, we'll use '`Phi(t)p(.)`'. Once **MARK** is

finished, go ahead and append the results from this run to the the results browser.

What you might see is that model 'Phi(t)p(.)' (representing model $\{\varphi_t p_.\}$) is listed first, and model 'Phi(t)p(t)' second, even though model 'Phi(t)p(t)' was actually run first. As you may recall from our discussions in Chapter 3, the model ordering is determined by a particular criterion (say, the AIC), and not necessarily the order in which the models were run.

Before we delve too deeply into the results of our analyses so far, let's finish our list of candidate models. We have model $\{\varphi_. p_t\}$ and model $\{\varphi_. p_.\}$ remaining. Let's start with model $\{\varphi_. p_t\}$ - constant survival, and time-dependent recapture probability. If you think about it for a few seconds, you'll realize that this model is essentially the 'reverse' of what we just did - constant survival and time-dependent recapture, instead of the other way around. So, you might guess that all you need to do is reverse the indexing in the survival and recapture PIMs. Correct again! Start with the survival PIM. Click in the first cell (upper left-hand corner), and then pull down the '**Initial**' menu and select '**Constant**', as you did previously. The survival PIM will change to a matrix of all '1's.

What about the recapture PIM? Again, click in the first cell. Since we're reusing the PIMs from our last analysis, the value of the first cell in the recapture PIM will be the number '7'. If we pull down the '**Initial**' menu and select '**Time**', we'd see the matrix change from all '7's to values from 7 to 12. Now, think about this for a minute. If we stop at this point, we'd be using the parameter index '1' for survival (constant survival), and indices 7 through 12 for recapture probability. What happened to indices 2 through 6?

In fact, nothing has really happened to them - but you don't know that yet. While it might make more sense to explicitly index the recapture PIM from 2 through 7, in fact, **MARK** will do this for you - but only during the numerical estimation itself. In fact, **MARK** more or less assumes that you've used '1' and '7 through 12' as the index values by mistake, and actually uses '1' and '2 through 7' when it does its calculations. Let's prove this to ourselves. Leave the PIMs as they are now - all '1's for survival, and '7 through 12' for recapture, and press the '**Run**' icon on the PIM toolbar. You'll be dumped into the '**Numerical Estimation**' setup window. For a title, we'll use 'Phi(.)p(t)'. Then, simply click on the '**OK to Run**' button. Append the results to the browser. Now, before looking at the browser itself, have another look at both the survival and recapture PIMs. What you'll see is that **MARK** has 'corrected' the PIM indexing for the recapture PIM, such that it is now '2 through 7', rather than '7 through 12'. Clever, eh? Yes, and no. It is nice that **MARK** does this for you, but you should **not** rely on software to do too much 'thinking' for you. It would have been better (albeit, somewhat slower) to simply index the recapture PIM correctly in the first place.

How would you do this? Aaah. . .this is where selecting the first cell in the PIM itself becomes important. Initially, the recapture PIM had all '7' values. You want to change it to time-dependent, but want the first value to be '2' (since the survival parameter index is '1'). To do this, simply change the value in the first cell of the recapture PIM from '7' to '2', and then select '**Time**' from the '**Initial**' menu. Let's put this into practice with the final model in our candidate model set - the constant survival and recapture model, $\{\varphi_. p_.\}$. For this model, the survival and recapture PIMs will be '1's and '2's, respectively.

begin sidebar

Why not 2's?

Why not use 2's for indexing survival, and 1's for recapture? In fact, **MARK** doesn't care at all - in **MARK**, the ordering or sequencing of PIM indexing is as arbitrary as which window you place where on the screen - it's entirely up to you. You would get the same 'results' using either '1' and '2' for survival and recapture, or '2' and '1', respectively.

end sidebar

In other words, for survival

1	1	1	1	1	1
	1	1	1	1	1
		1	1	1	1
			1	1	1
				1	1
					1

and for recapture

2	2	2	2	2	2
	2	2	2	2	2
		2	2	2	2
			2	2	2
				2	2
					2

One simple way to remember what the triangular matrix is telling you is to remember that time moves left to right, and cohort from top to bottom. If the numbers (indices) change in value from left to right, then survival changes with time. If they change from top to bottom, they change over cohort. Of course, the indices can change in either one or both directions simultaneously.

Once the PIMs are set, run the model (we'll use '`Phi(.)p(.)`' for the model name), and append the results to the browser. You now have 4 different model results in the browser, corresponding to each of the 4 models we're interested in. Of course, there are **many** other models we could have tried, but at this stage, we simply want to get comfortable building models in **MARK**. As we'll discuss shortly, the selection of the candidate set of models is crucial to our task. For now though, let's just consider these 4 as representative of models we have an interest in. Let's start by looking at the results browser itself.

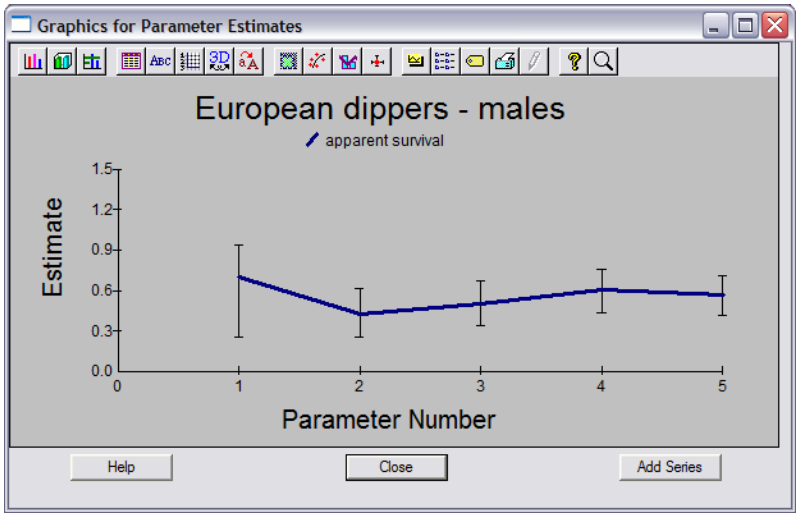
Results Browser: Live Recaptures (CJS)

Model	AICc	Delta AICc	AICc Weight	Model Likelihood	No. Par.	Deviance
{phi(.)p(.)}	322.5527	0.0000	0.96003	1.0000	2	41.8147
{phi(t)p(.)}	330.0567	7.5040	0.02253	0.0235	7	38.8147
{phi(.)p(t)}	330.6794	8.1267	0.01650	0.0172	7	39.4374
{phi(t)p(t)}	336.4343	13.8816	0.00093	0.0010	11	36.4013

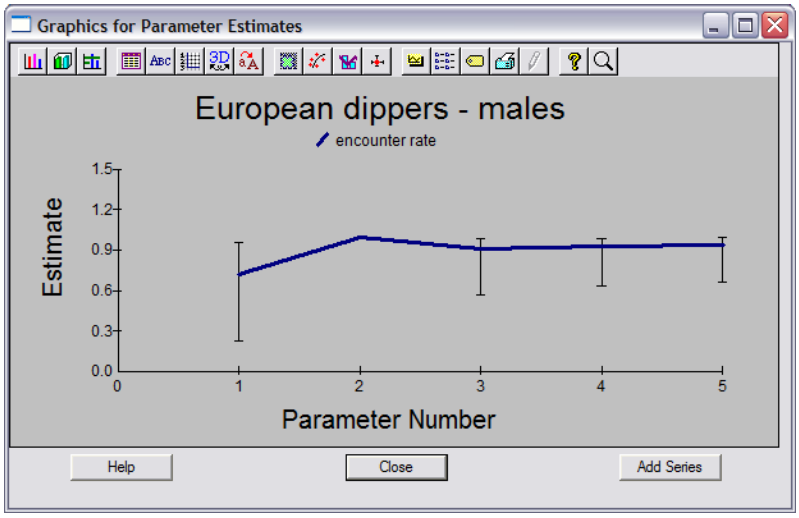
We see that the 4 models are listed, in order of ascending AIC, ranging from 322.55 for model $\{\varphi_{\cdot} p_{\cdot}\}$ to 336.43 for model $\{\varphi_t p_t\}$.

Before we evaluate the results from our 4 models, it is often a good starting point to take the estimates from the most fully parameterized model, and plot them (**MARK** has some basic 'built-in' plotting capabilities - simply click the 'line graph' icon in the browser toolbar, and go from there. Fairly self-explanatory). Often, a sense of the underlying model structure is revealed by examination of the estimates from the most parameterized model. The reason is fairly straightforward - the more

parameters in the model, the better the fit (smaller the deviance - more on model deviance later on). As we will discuss shortly, this does not necessarily mean that it is the best model, merely the one that fits the best (this is a crucial distinction). However, the most parameterized model generally gives the most useful 'visual' representation of the pattern of variation in survival and recapture. In the case of our 4 models, the most parameterized is model $\{\varphi_t p_t\}$ - the CJS model. The parameter estimates (with 95% CI) for φ and p are plotted below - first, the estimated apparent survival probabilities ($\hat{\varphi}_i$)



Then, the estimated recapture probabilities ($\hat{p}_i$)



Note that in these figures we do not include all 6 estimates that **MARK** derives for both survival and recapture. Why? As it turns out, the final estimate for both survival and recapture is 0.7638. Is this just a coincidence? No! In fact, what **MARK** has done is estimate the square-root of the combined probability $\varphi_6 p_7$ (which Lebreton *et al.* (1992) refer to as β_7). For the time-dependent CJS model, the components of this product are not individually identifiable - without further information, we cannot separately estimate survival from recapture - we can only estimate the square-root of the product. We shall discuss

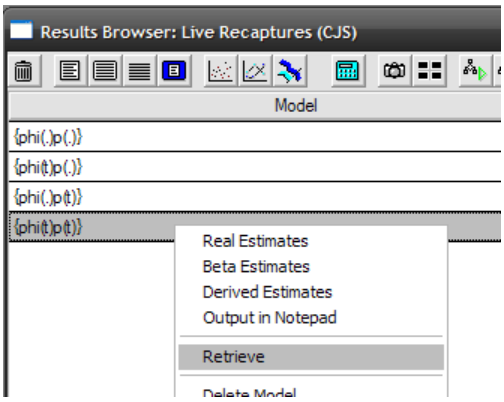
this again in more detail later on. Since this $\varphi_6 p_7$ product term is not comparable to either survival or recapture probabilities separately, it is excluded from our plots. Of course, if you've looked at the output listing already, you may have 'seen' that parameters 6 and 12 are not separately identifiable. However, as we've mentioned before, we do not favor unquestioning reliance on the ability of the software (be it **MARK** or any other application) to determine the number of estimable parameters - you should first develop an understanding of how it is done from first principals. This is covered in the Addendum section at the end of this chapter (we have placed it there to minimize disruption to the flow of the material on building models. However, you are strongly urged to read it carefully, at some point).

4.2. A quicker way to build models - the PIM chart

In the preceding example, we started our model building by opening the PIMs for both survival and recapture (the two primary parameters in the live encounter analysis of the male Dipper data). We modified the PIMs to set the underlying parameter structure for our models. However, at this point, we want to show you another, more efficient way to do the same thing, by introducing one of the 'whiz-bang' (from the Latin) features of **MARK** - the *Parameter Index Chart* (hereafter referred to as the 'PIM chart'). Explaining what it is is probably most effectively done by letting you have a look. We'll do so by considering two different numerical examples - one involving a single group of marked individuals, and the second involving multiple groups of marked individuals. Not only is the situation involving multiple groups quite common, but some of the notable advantages of using the PIM chart to speed model construction are even more obvious for analyses involving multiple groups of marked individuals.

4.2.1. PIM charts and single groups - Dipper re-visited

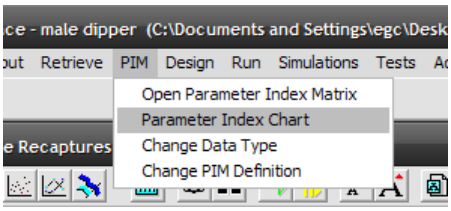
Open up the male Dipper data analysis - there should be 4 models in the results browser. We're going to replicate these 4 models again, this time using the PIM chart, rather than manually modifying the individual PIMs. We'll start with model $\{\varphi_t p_t\}$. Simply find that model in the results browser, right-click it and select '**`Retrieve`**' from the sub-menu. This will make the underlying PIM structure for this model active (you can always check this by opening up each of the individual PIMs and having a look).



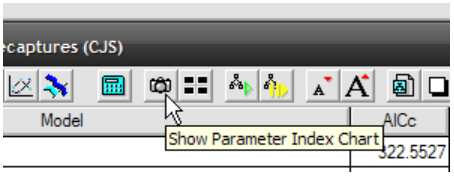
Recall that for model $\{\varphi_t p_t\}$, there are 6 intervals for survival, and 6 occasions for recapture - so, in theory, 12 total parameters that could be estimated: $1 \rightarrow 6$ for survival, and $7 \rightarrow 12$ for recapture (although we remember that for the fully time-dependent model, the final survival and recapture

parameters are confounded - such that only 11 total parameters will be estimated). Recall that at this point, we're simply setting the underlying parameter structure for our models.

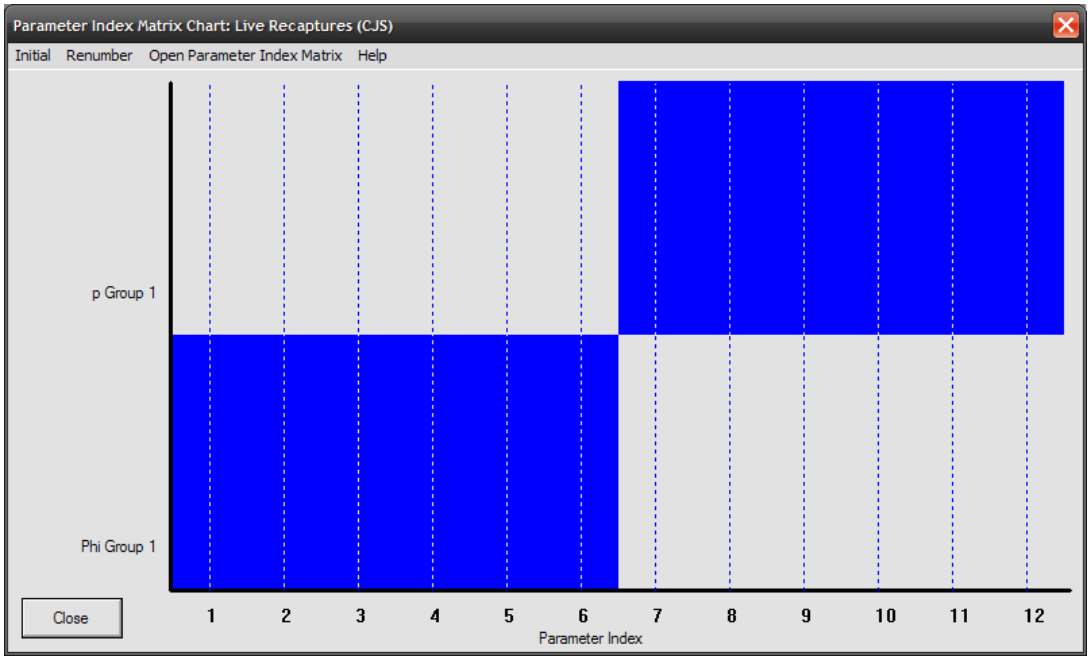
Now, let's have a look at this thing called the PIM chart. You can either (i) select '**`Parameter Index Chart`**' from the PIM menu



or (ii) click the appropriate icon in the main toolbar (for the PIM chart, this is the icon that looks like a 'camera', more or less).



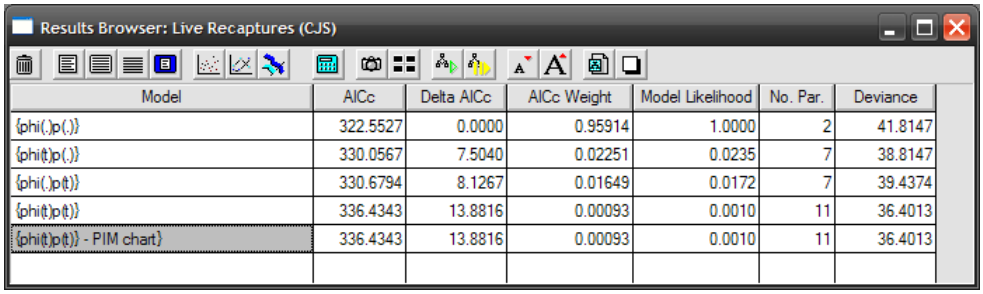
Go ahead and open up the PIM chart for model $\{\varphi_t p_t\}$.



Zowie! OK. . .now, what is it? The PIM chart is a simple, yet useful visual tool for looking at the parameter indexing **MARK** uses for the various groups and parameters in the current model (in our case, model $\{\varphi_t p_t\}$). What you can see from the chart is that there are 2 main 'groupings' of parameters

for this model: survival (φ) respectively, and recapture (p), respectively. Along the bottom axis is the parameter index itself, and along the vertical axis are the parameters. So, in this example, parameters 1 to 6 refer to the survival probabilities, and 7 to 12 correspond to the recapture parameters.

Now, at this point we haven't changed anything to the parameter structure - the PIM chart simply reflects the structure of the current model. You can confirm that in fact nothing has changed by running this model, and adding the result to the browser. Make the title for the model '`Phi(t)p(t) - PIM chart`' - adding the extra label will help you identify which models in the browser were created using the PIM chart.



Model	AICc	Delta AICc	AICc Weight	Model Likelihood	No. Par.	Deviance
{phi(.)p(.)}	322.5527	0.0000	0.95914	1.0000	2	41.8147
{phi(t)p(.)}	330.0567	7.5040	0.02251	0.0235	7	38.8147
{phi(.)p(t)}	330.6794	8.1267	0.01649	0.0172	7	39.4374
{phi(t)p(t)}	336.4343	13.8816	0.00093	0.0010	11	36.4013
{phi(t)p(t)} - PIM chart}	336.4343	13.8816	0.00093	0.0010	11	36.4013

As you can see, the results for model '`Phi(t)p(t) - PIM chart`' are identical to those for model '`Phi(t)p(t)`'.

OK, so far, the PIM chart hasn't really done much for us, other than provide a convenient visual representation of the underlying parameter structure for our model. In fact, the greatest utility of the PIM chart is the ease with which you can use it to build other models. We'll demonstrate that now. Let's consider building model $\{\varphi_t p_.\}$. How can we build this model using the PIM chart? Recall that for this model, the underlying parameter structure for survival should look like

1	2	3	4	5	6
	2	3	4	5	6
		3	4	5	6
			4	5	6
				5	6
					6

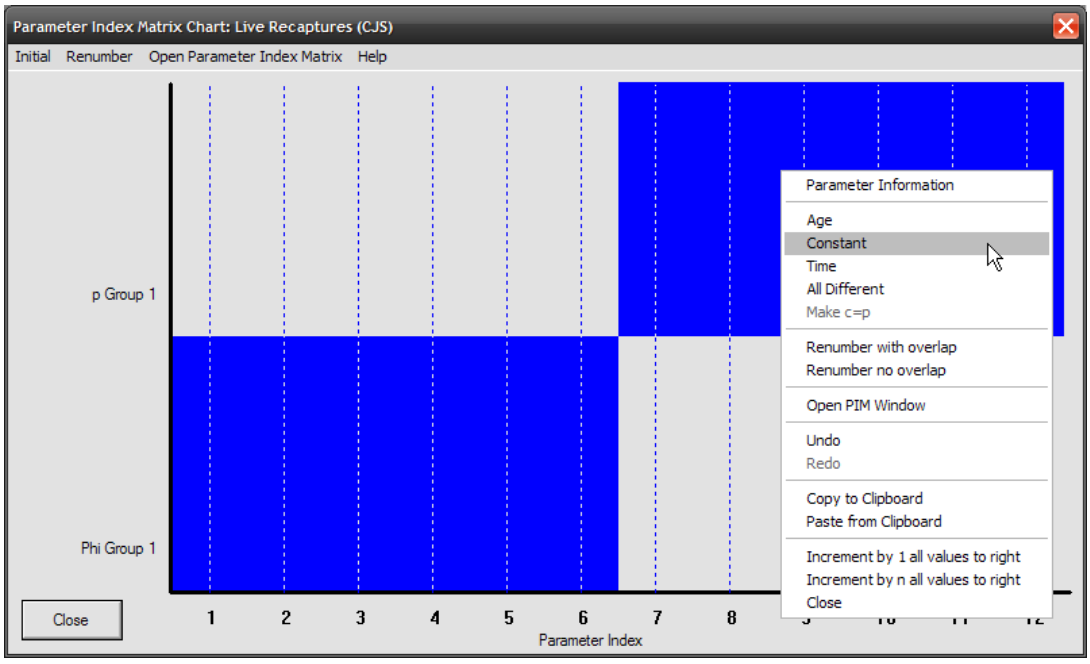
and for recapture

7	7	7	7	7	7
	7	7	7	7	7
		7	7	7	7
			7	7	7
				7	7
					7

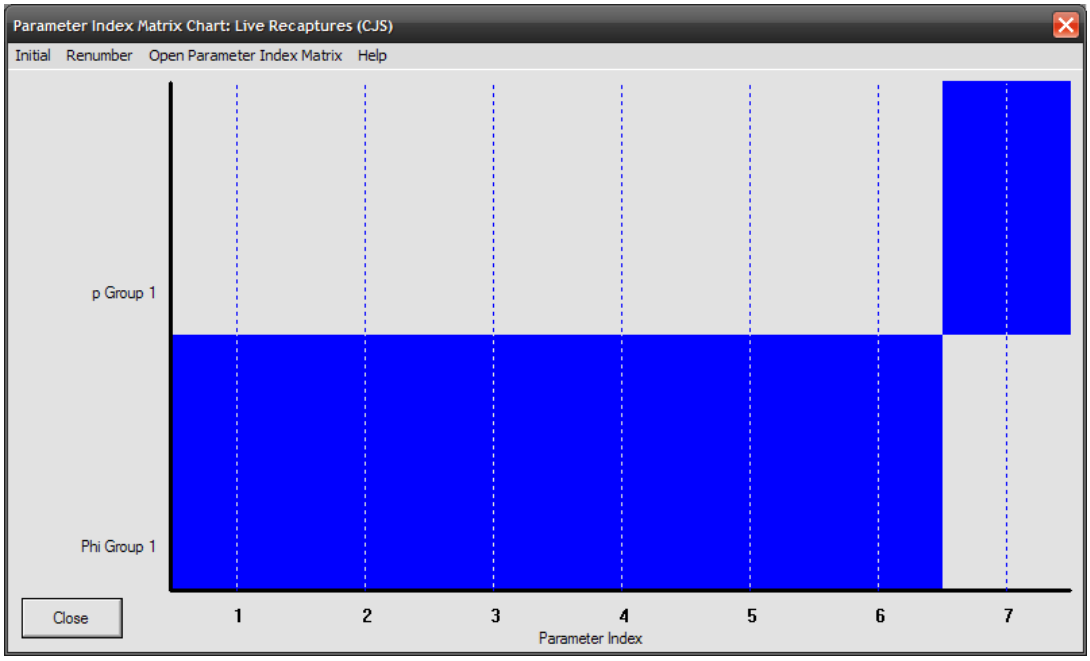
However, the recapture PIM for our current model $\{\varphi_t p_t\}$ has the structure

7	8	9	10	11	12
	8	9	10	11	12
		9	10	11	12
			10	11	12
				11	12
					12

So, we want to change the recapture PIM from time-dependent (index values $7 \rightarrow 12$) to time-invariant (constant; index values 7 only for all occasions). How do we do this using the PIM chart? Easy! Simply open up the PIM chart, and right click on the 'blue-box' corresponding to the recapture parameter. This will spawn a sub-menu which list various options - the option we want to select is '**`Constant`**' (shown below):

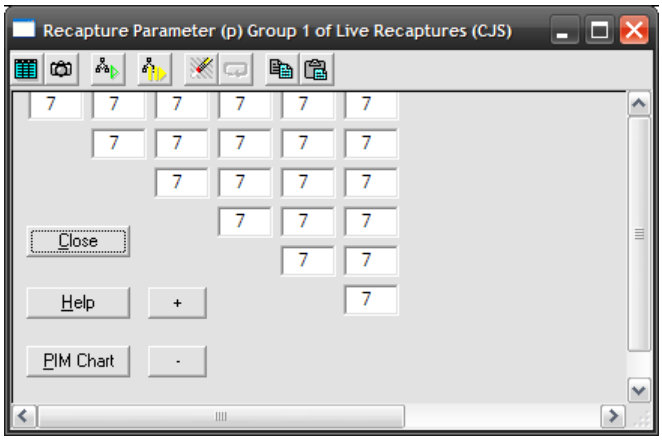


What this will do is change the parameter structure for the selected 'blue box' (which in this case represents the recapture parameter) and change it from the current structure (in this case, time-dependent) to constant. Go ahead and select '**`Constant`**' – the results are shown at the top of the next page:



Now, the right-most blue box has only one parameter index - '7'. The size of the box has changed to reflect that we've gone from full time-dependence (6 parameters wide) to a constant 'dot' model (1 parameter wide).

We can confirm this by opening up the recapture PIM.

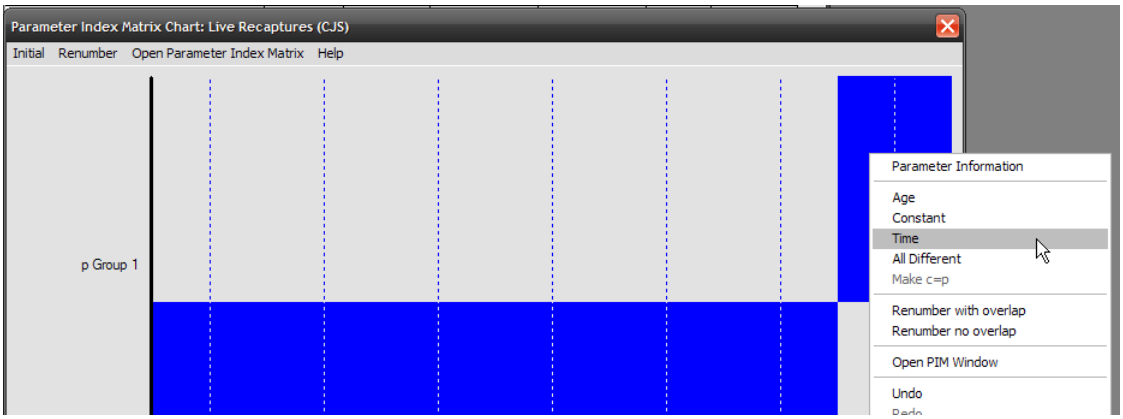


As expected, it consists entirely of the number '7'.

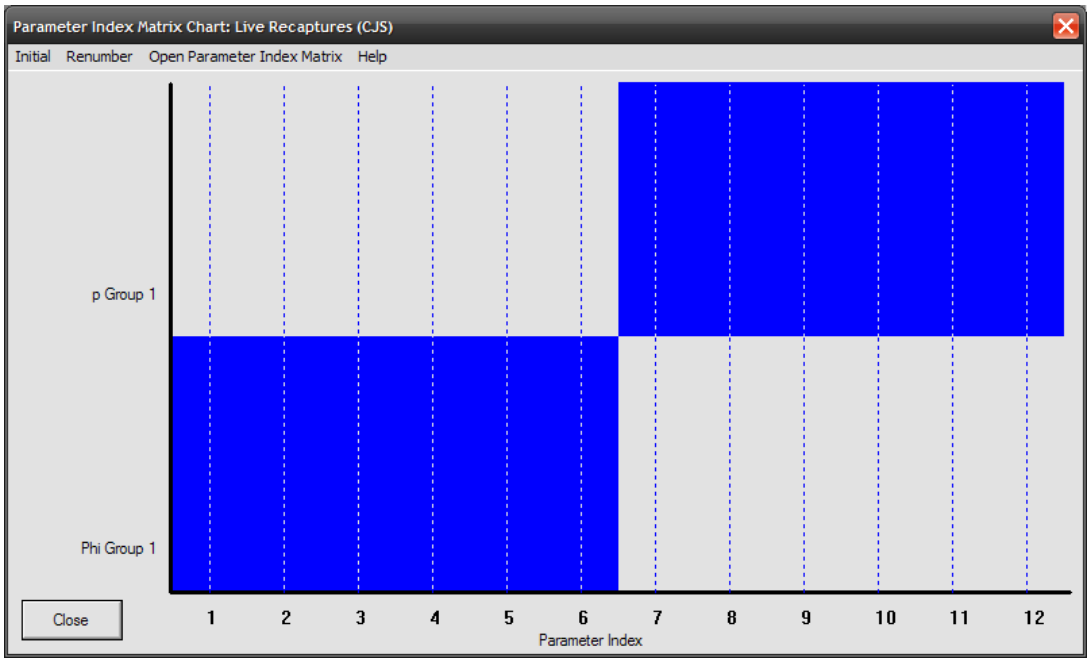
Now, wasn't that fast? To build model $\{\varphi_t p_\cdot\}$ from model $\{\varphi_t p_t\}$, all we did was (i) open up the PIM chart for model $\{\varphi_t p_t\}$, (ii) right-click the 'blue box' corresponding to the recapture parameter, and (iii) select constant.

Go ahead and run this model - label it '`phi(t)p(.) - PIM chart`', and add the results to the browser. The results should be identical to those from model '`phi(t)p(.)`', which we fit by manually modifying the parameter-specific PIMs.

Now, what about model $\{\varphi_{\cdot} p_t\}$? At this point we could either go back into the browser, retrieve model $\{\varphi_t p_t\}$, bring up the PIM chart, and repeat the steps we just took, except right-clicking on the survival 'blue box', rather than the recapture 'blue box'. Alternatively, we could simply bring up the PIM chart for the model we just fit $\{\varphi_t p_{\cdot}\}$ and modify that. We'll use the second approach. Go ahead and bring back up the PIM chart. Now, we're going to right-click on the recapture 'blue-box', and change it from constant to time-dependent:

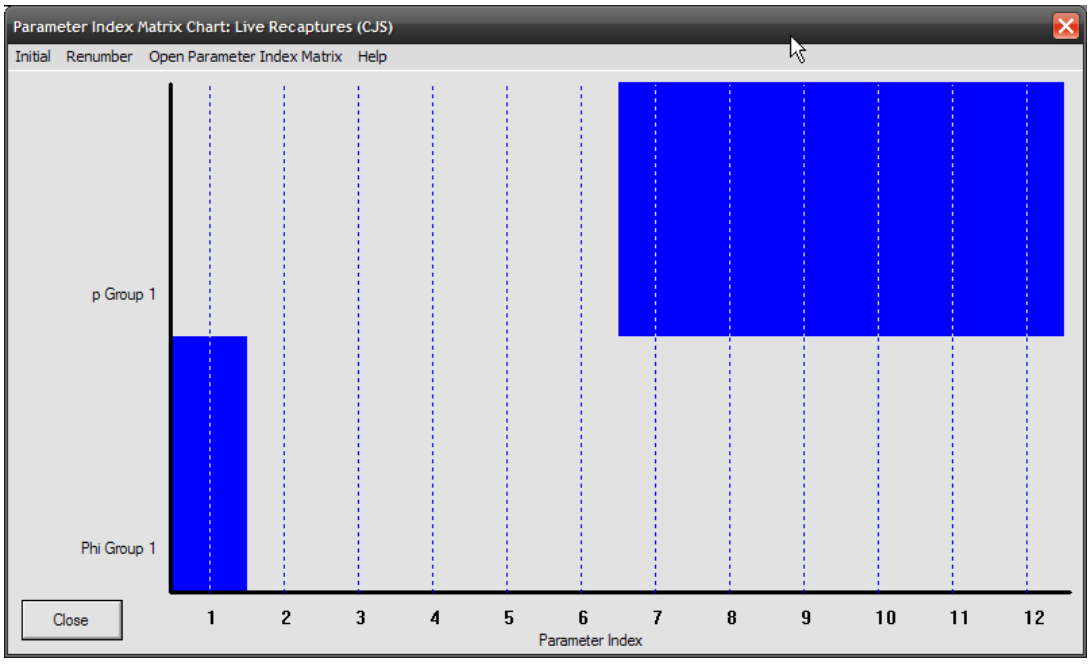


This will generate a PIM chart that looks like



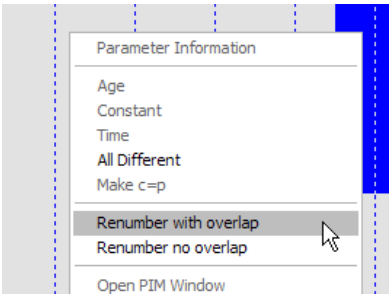
Recognize it? You should - it's the PIM chart corresponding to the model we started with $\{\varphi_t p_t\}$ - which has 6 survival parameters, and 6 recapture parameters.

Now, right-click on the survival 'blue-box', and select '**Constant**'. Remember, we're trying to build model $\{\varphi_{\cdot} p_t\}$.

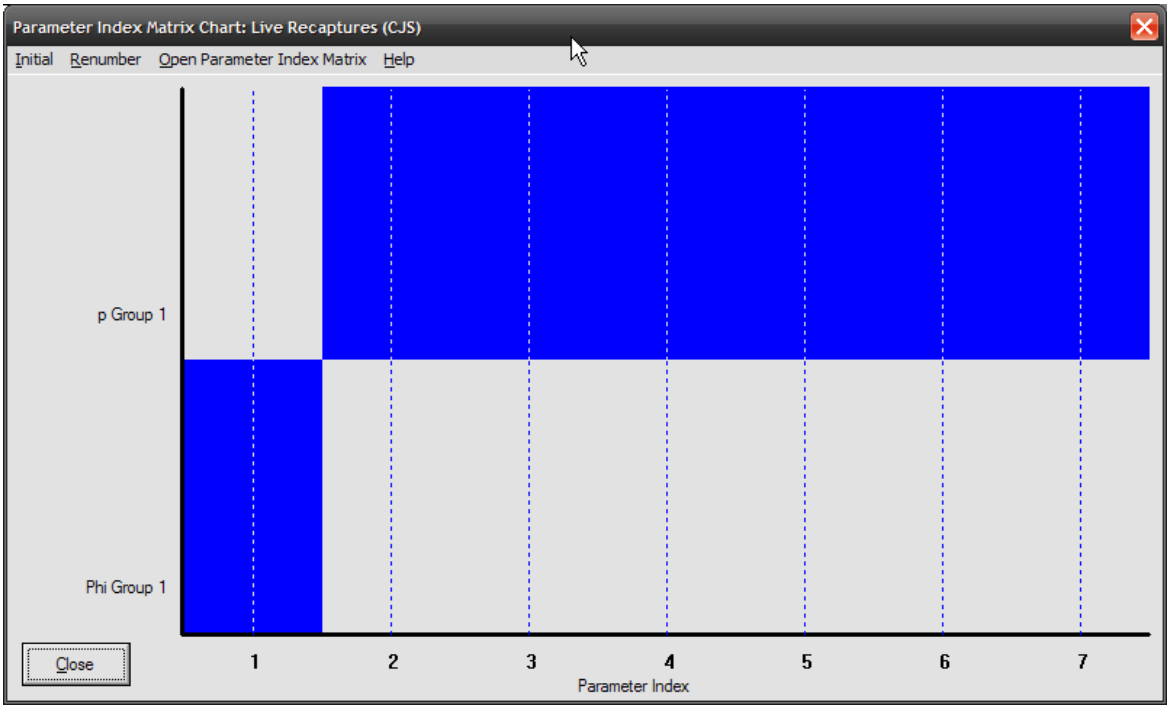


So, a couple of things to notice here. First, as intended, the 'blue box' for the survival parameter has 'shrunk', reflecting the fact that we've gone from time-dependent (parameters $1 \rightarrow 6$) to constant (parameter 1).

But, we also notice there is a substantial 'gap' between the two 'blue-boxes'. Parameter index values $2 \rightarrow 6$ don't correspond to anything. We want to eliminate the gap (i.e., remove the meaningless index values). You could do this in one of two ways. First, the PIM chart lets you manually 'drag' the 'blue-boxes' around. So, you could left-click the recapture 'blue-box' and, while holding down the left mouse button, drag the recapture blue box to the left, so that the left-most edge of the box corresponds to parameter index 2. Try it, it's pretty slick. Alternatively, you can right-click anywhere on the PIM chart, and select either of the '**Renumber**' options you are given (the distinction between the two will become obvious in our next worked example):



Doing so will cause the PIM chart to change (shown at the top of the next page) – the gap between the two 'blue boxes' will be eliminated, and the structure will now reflect model $\{\varphi_{\cdot} p_t\}$.



Go ahead and run the model, label it '`phi(.)p(t) - PIM chart`', and add the results to the browser. Again, they should be identical to the results from fitting model '`phi(.)p(t)`' built by modifying the individual PIMs. Again, using the PIM chart is much faster.

As a final test, try fitting model $\{\varphi\ p\ \}$. You'll know you've done it correctly if the results match those for '`phi(.)p(.)`' already in the browser.

4.2.2. PIM charts and multiple groups

This second worked example involves some data from two different nesting colonies of the swift (*Apus apus*), a small insectivorous bird. In addition to providing an opportunity to demonstrate the use of the PIM chart, this data set also introduces the general concept of comparing groups (an extremely common type of analysis you're likely to run into routinely). In fact, as we will see, it involves only slightly more work than the model comparisons we saw in the Dipper example we considered in the first part of this chapter.

The data consist of live encounter data collected over an 8 year study of 2 different swift colonies in southern France. One of the two colonies was believed to be of 'poorer' quality than the other colony for a variety of reasons, and the purpose of the study was to determine if these perceived differences between the two colonies (hereafter, **P** - 'poor', and **G** - 'good') were reflected in differences in either survival or recapture probability. The data for both the **P** and **G** colonies are both in `AA.INP` - the encounter frequencies are tabulated for the 'poor' and 'good' colonies, respectively. In this example, we will analyze the data in terms of the following 2 factors: colony (**G** or **P**), and time. Thus, this example is very similar to the Dipper example, except that we have added one more factor, colony. As such, the number of possible models is increased from $2^2 = 4$ models to $4^2 = 16$ models – survival and recapture could vary with colony, time or both. The candidate set of models is shown at the top of the next page.

$$\begin{array}{cccc} \{\varphi_{c*t}p_{c*t}\} & \{\varphi_{c}p_{c*t}\} & \{\varphi_{t}p_{c*t}\} & \{\varphi_{.}p_{c*t}\} \\ \{\varphi_{c*t}p_{c}\} & \{\varphi_{c}p_{c}\} & \{\varphi_{t}p_{c}\} & \{\varphi_{.}p_{c}\} \\ \{\varphi_{c*t}p_{t}\} & \{\varphi_{c}p_{t}\} & \{\varphi_{t}p_{t}\} & \{\varphi_{.}p_{t}\} \\ \{\varphi_{c*t}p_{.}\} & \{\varphi_{c}p_{.}\} & \{\varphi_{t}p_{.}\} & \{\varphi_{.}p_{.}\} \end{array}$$

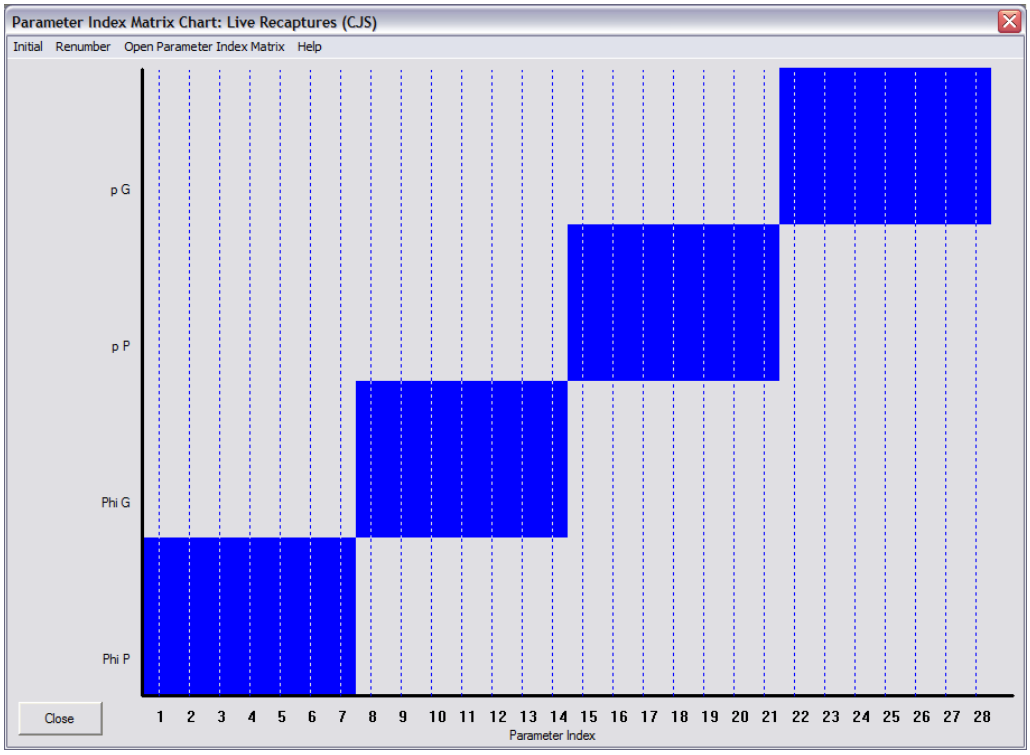
With an increasing number of factors, the number of possible models that may need to be tested increases geometrically. Here we have an 8 year study, considering only 2 primary factors (colony and time), and there are at least 16 possible models to test (in fact, we will see in subsequent chapters there are potentially many more).

Two points to make before we go any further. First, we should make sure you understand the syntax of the model representations in the preceding table (which follow the approach recommended in Lebreton *et al*. 1992). Remember, that the subscripts for the two parameters (φ and p) reflect the structure of the model. The most general model in the table is model $\{\varphi_{c*t}p_{c*t}\}$. The '`c*t`' subscript means we have a 'full' model (for both survival and recapture), including both the main effects (`colony` and `time`) and the interaction of the two (i.e., '`c*t` = `c + t + c.t + error`). By 'interaction', we are referring to the statistical meaning of the word - that `colony` and `time` interact, such that the relationship between survival or recapture and time can differ depending upon the colony (or conversely, the relationship between survival or recapture and `colony` can differ depending upon the time interval). This model is the most general, since any of the other models listed can be derived by removing one or more factors (a simple comparison of the 'complexity' of the subscripting for both survival and recapture among the various models will confirm this).

Second, by now you've no doubt noticed that we highlighted the word 'possible' repeatedly. We did so for a reason - to foreshadow discussion of 'model selection' and 'model uncertainty', presented later in this chapter. For the moment, we'll assume that we are particularly interested in whether or not there are differences in survival between the 2 colonies. We'll assume that there are no differences in recapture probability between the colonies. These assumptions are reflected in our 'candidate model set', which is a subset of the table presented above:

$$\begin{array}{ccc} \{\varphi_{c*t}p_{t}\} & \{\varphi_{t}p_{t}\} & \{\varphi_{c}p_{t}\} \\ \{\varphi_{c*t}p_{.}\} & \{\varphi_{t}p_{.}\} & \{\varphi_{c}p_{.}\} \end{array}$$

Note that this candidate model set reflects some 'prior thinking' about the data set, the analysis, the biology - different investigators might come up with different candidate model sets. However, for the moment, we'll use this particular candidate model set, and proceed to analyze the data. We will start by fitting the data to the most *general* approximating model in the model set $\{\varphi_{c*t}p_{t}\}$. It is the most general, because it has the most parameters of all the models we will consider. Start program **MARK**, and start a '**New**' project (i.e., from the '**File**' menu, select '**New**'). Pull in the data from `AA.INP` (2 groups, 8 occasions - the first frequency column represents the 'poor' colony, while the second frequency column represents the 'good' colony). Next, either pull down the '**PIM**' menu, and select '**Parameter Index Chart**', or select the PIM chart icon on the toolbar. The resulting (default) PIM chart corresponds to model $\{\varphi_{c*t}p_{c*t}\}$ is shown at the top of the next page. Again, what you can see from the chart is that there are 4 main 'groupings' of parameters for this model: survival for good and poor colonies respectively, and recapture for the good and poor colonies, respectively. Along the bottom index is the parameter index itself, and along the vertical axis are the parameters and group labels. So, in this example, parameters 1 to 7 refer to the survival probabilities for the poor colony, 8 to 14 correspond to the survival parameters for the good colony, and so on. As mentioned in the Dipper example we just completed, the PIM chart allows you to quickly determine the structure of your model, in terms of the parameters indices.



However, before we proceed, think back for a moment to our candidate model set. In the model set, the most general model we want to fit is model $\{\varphi_{c*t}p_t\}$. However, the default model **MARK** starts with is always the fully structured model, in this case, model $\{\varphi_{c*t}p_{c*t}\}$. So, as a first step, we need to reconfigure the default model from $\{\varphi_{c*t}p_{c*t}\}$ to $\{\varphi_{c*t}p_t\}$. This involves changing the parameter structure for the recapture parameters, by eliminating the colony effect.

This should be reasonably straightforward. We have 8 occasions, and 2 groups. Thus, for a given group, we have 7 survival intervals, and 7 recapture occasions. For model $\{\varphi_{c*t}p_{c*t}\}$, the parameters would be numbered:

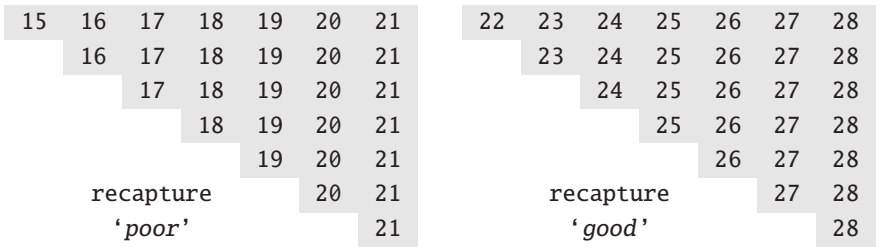
survival		*recapture*	
poor	good	poor	good
$1 \rightarrow 7$	$8 \rightarrow 14$	$15 \rightarrow 21$	$22 \rightarrow 28$

In other words, the PIMs would look like the following for survival:

1	2	3	4	5	6	7
	2	3	4	5	6	7
		3	4	5	6	7
			4	5	6	7
				5	6	7
		survival			6	7
		'poor'				7

8	9	10	11	12	13	14
	9	10	11	12	13	14
		10	11	12	13	14
			11	12	13	14
				12	13	14
		survival			13	14
		'good'				14

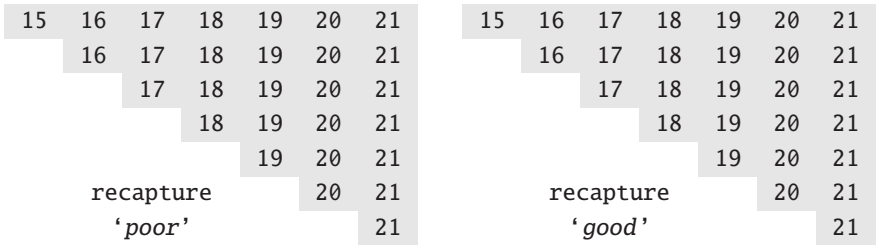
and for recapture



Now, what we want to do is modify this structure to reflect model $\{\varphi_{c*t}p_t\}$ - in other words, we want to change the recapture PIMs so that they are the same between the two groups (the two colonies):

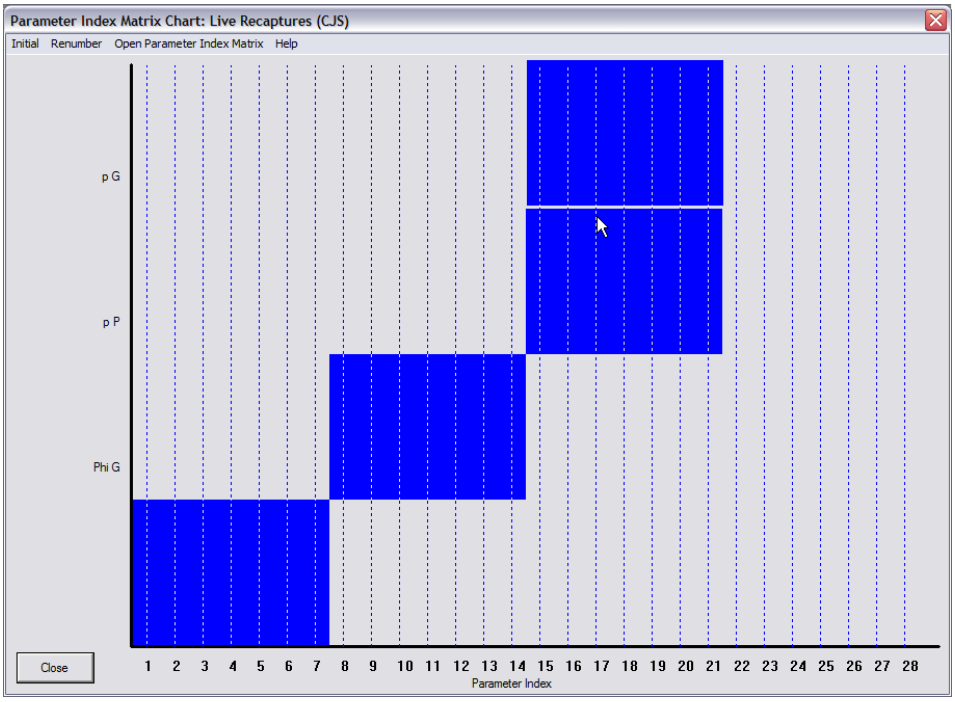
survival		*recapture*	
poor	good	poor	good
$1 \rightarrow 7$	$8 \rightarrow 14$	$15 \rightarrow 21$	$15 \rightarrow 21$

The recapture PIMs would now look like:



While we could do this 'manually', by modifying the indexing for each individual PIM, **MARK** lets you accomplish this in a faster, more elegant way - by modifying the PIM chart directly. How? By simply selecting (left-click with the mouse) the 'good' colony 'blue box' in the PIM chart, and while holding down the left mouse button, dragging it to the left, so that it lines up with the recapture 'blue box' for the 'poor' colony, then releasing the mouse button (as shown at the top of the next page). Compare this PIM chart with the original one.

Next, look at the PIMs – you'll see that the recapture PIMs are now identical for both groups, indexed from $15 \rightarrow 21$, just as they should be. Now isn't that easy? We're now ready to run our general, starting model $\{\varphi_{c*t}p_t\}$. Go ahead and run it, calling it model '`Phi(c*t)p(t)`'. Add the results to the browser. The model deviance is 107.563, with 20 estimated parameters (6 survival parameters for the 'poor' colony, 6 survival parameters for the 'good' colony, 6 recapture probabilities (the same for both colonies), and 2 β-terms (one for each colony). Make sure you understand the parameter counting!



Now, we simply run the other models in the candidate model set: $\{\varphi_{c*t}p_{\cdot}\}$, $\{\varphi_t p_t\}$, $\{\varphi_c p_t\}$, $\{\varphi_t p_{\cdot}\}$ and $\{\varphi_c p_{\cdot}\}$. To reinforce your understanding of manipulating the PIM chart, and to demonstrate at least one other nifty trick with the PIM chart, we'll start with model $\{\varphi_t p_t\}$. For this model, the PIM structure would be:

For survival

1	2	3	4	5	6	7
	2	3	4	5	6	7
		3	4	5	6	7
			4	5	6	7
				5	6	7
		survival			6	7
		'poor'				7

1	2	3	4	5	6	7
	2	3	4	5	6	7
		3	4	5	6	7
			4	5	6	7
				5	6	7
		survival			6	7
		'good'				7

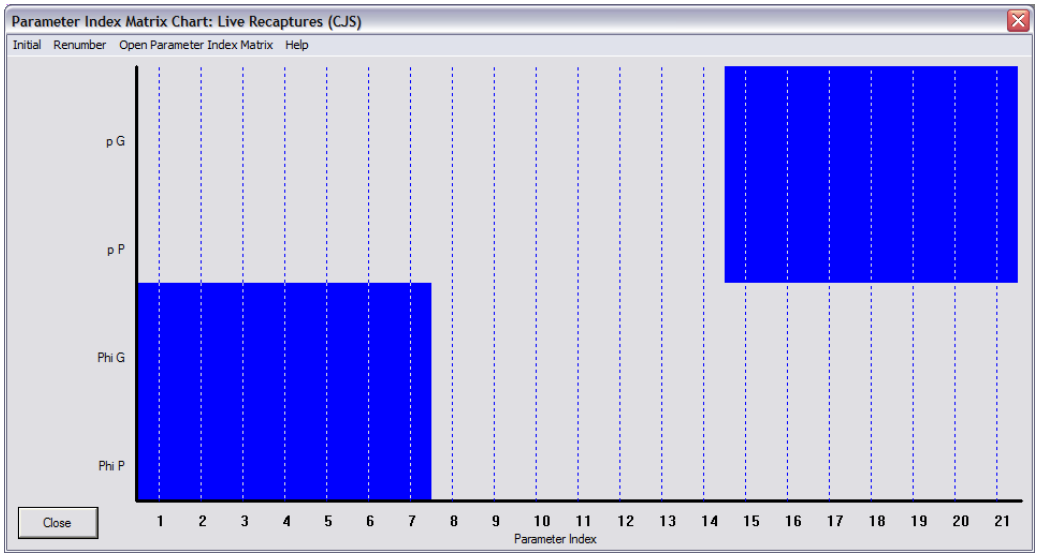
and for recapture

8	9	10	11	12	13	14
	9	10	11	12	13	14
		10	11	12	13	14
			11	12	13	14
				12	13	14
		recapture			13	14
		'poor'				14

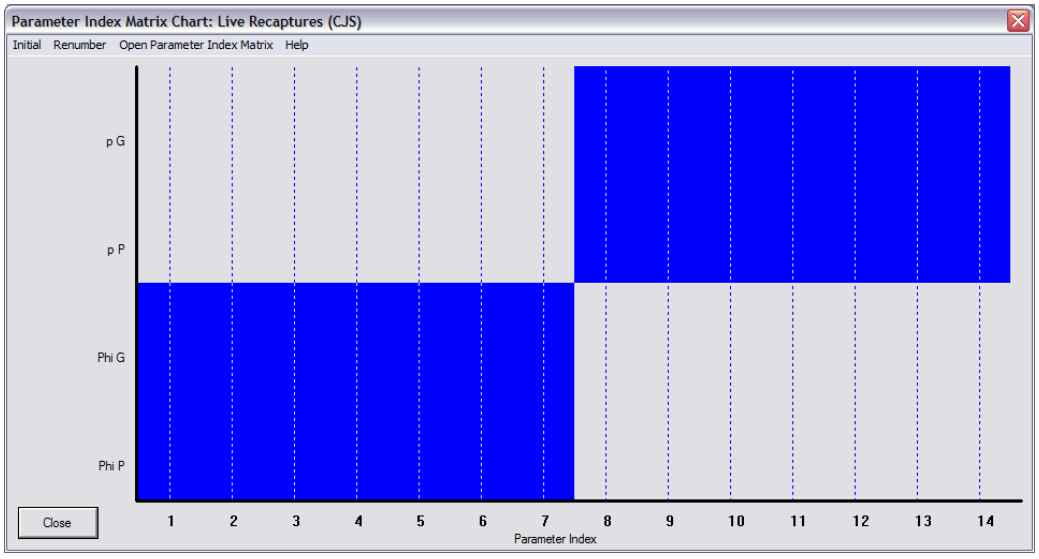
8	9	10	11	12	13	14
	9	10	11	12	13	14
		10	11	12	13	14
			11	12	13	14
				12	13	14
		recapture			13	14
		'good'				14

Now, if you understood our first attempt with manipulating the PIM chart, you might guess (correctly) that what you need to do is 'make the blue boxes for the survival parameters line up'. However,

if you look at the chart, you see you could do this by grabbing the 'blue box' for survival for the poor colony and dragging it to the right (under the box for the good colony), or, in reverse, grabbing the box for the good colony, and dragging it to the left. We'll use the latter approach, because we want to point out another feature of the PIM chart that is worth noting. Here is the PIM chart.



Notice that now there is a gap between the 'stacked blue boxes' for the survival and recapture parameters – survival is indexed from 1 → 7, while recapture is indexed from 15 → 21. The index values for 8 → 13 don't correspond to any parameter. We want to eliminate the gap (i.e., remove the meaningless index values). You could do this manually, simply by dragging both recapture blue boxes to the left. Or, you could do this by right-clicking anywhere in the PIM chart. You'll be presented with a menu, which has '**`Renumber with overlap`**' as one of the options. '**`Renumber with overlap`**' means (basically), renumber to eliminate any gaps, but allow for the blue boxes for some parameters to overlap each other'. If you select the '**`renumber with overlap`**' option, the PIM chart will change to:



Pretty slick, eh? This corresponds to model $\{\varphi_t p_t\}$. Confirm this for yourself by checking the 4 individual PIMs (in fact, this is always a good idea, until you're 100% comfortable with **MARK**). Once you're sure you have the right model, go ahead and run it - call it '`phi(t)p(t)`', and add the results to the browser. This model has a much smaller AIC value than our starting model, although it has a larger deviance (which alone suggests that the time-dependent model does not fit the data as well as the more general model which included colony effects). We'll defer discussion/interpretation of these 'model fitting' considerations to the next section.

There are still several other models in our candidate model set. Go ahead and run them – here are the results for all of the models in the candidate model set:

Results Browser: Live Recaptures (CJS)

Model	AICc	Delta AICc	AICc Weight	Model Likelihood	No. Par.	Deviance	-2Log(L)
{phi(c)p(t)}	369.8080	0.0000	0.85650	1.0000	9	111.6644	350.8705
{phi(c)p(.)}	373.5263	3.7183	0.13345	0.1558	3	128.1990	367.4051
{phi(t)p(.)}	379.0123	9.2043	0.00859	0.0100	8	123.0601	362.2662
{phi(t)p(t)}	382.8807	13.0727	0.00124	0.0014	13	115.7384	354.9446
{phi(c*t)p(.)}	386.4962	16.6882	0.00020	0.0002	15	114.7094	353.9155
{phi(c*t)p(t)}	391.4103	21.6023	0.00002	0.0000	20	107.5633	346.7694

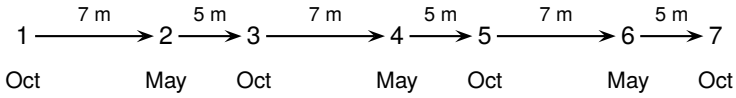
If your values for deviance and so on match those shown here, then that's a good clue that you've managed to build the models successfully (we'll be explaining what the various columns in the results browser are shortly).

begin sidebar

specifying and modeling uneven time-intervals between sampling occasions

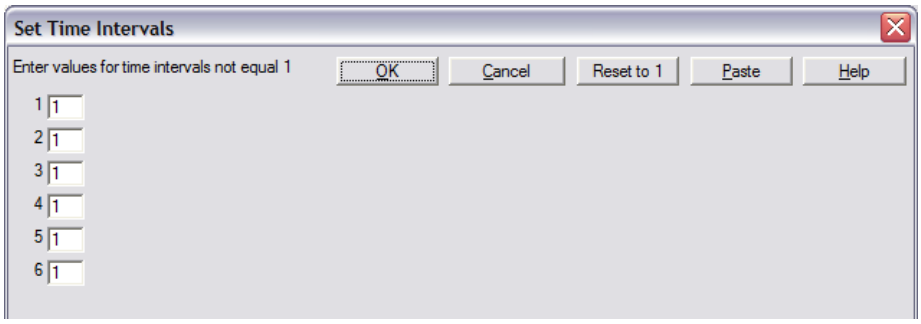
In the preceding, we have implicitly assumed that the sampling interval between encounter occasions is identical throughout the course of the study (e.g., sampling every 12 months, or every month, or every week). But, in practice, it is not uncommon for the time interval between occasions to vary - either by design, or because of 'logistical constraints'. This has clear implications for how you analyze your data.

Consider the following example (contained in `interval.inp`), where you sample a population each October, and again each May (i.e., two samples within a year, with different time intervals between samples; October→ May (7 months), and May→ October (5 months), for 7 occasions (assume the first sampling occasion is in October). Thus, the sampling intervals over the course of the study are



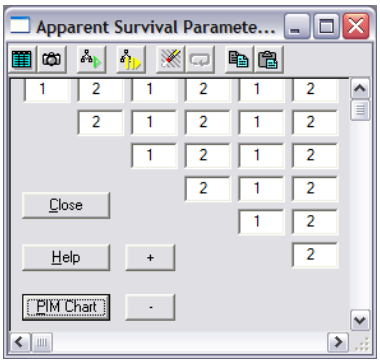
Suppose the 'monthly' survival probability for each month is 0.95 (this was the value used to simulate the data – the recapture probability in the simulation was held constant at $p = 0.80$ at each occasion). Thus, the expected 'seasonal' survival probability for the May→ October season is $0.95^5 = 0.7738$, and $0.95^7 = 0.6983$ for the October → May season; in other words, the same *monthly* survival between seasons, but different expected *seasonal* survival probabilities. But, more importantly, since the monthly survival probability is the same, then if the seasons were the same length (say, both 6 months long), then we would expect that seasonal survival for both seasons would be the same, and that the best, most parsimonious model would likely be one that constrained survival to be the same between seasons.

But, what happens if you fit a model to these data where survival is constrained to be the same between seasons, without correctly specifying the different time intervals between sampling occasions? Start **MARK**, and read in the file `interval.inp`. The simulated data represent a live mark-encounter study, which is the default data type in **MARK**. We specify 7 sampling occasions. If you click the button to the right of where you specify the number of encounter occasions, you'll see that **MARK** defaults to a common, constant interval of '1' time unit between each successive sampling occasion:



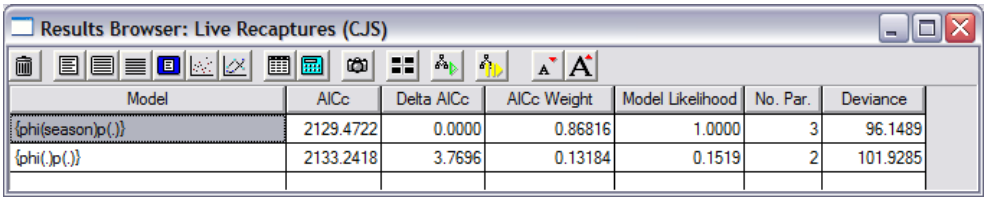
Of course, we know that for this example, these default intervals are incorrect, but to demonstrate what happens if you don't correctly specify the time interval we'll accept the default values of '1' between sampling occasions. We'll fit 2 models to these data: model $\{\varphi_{.}p_{.}\}$, and model $\{\varphi_{(season)}p_{.}\}$, where the second model assumes there is a different survival probability between seasons (but that within season, the estimated survival probability is constant among years). How do we build model $\{\varphi_{(season)}p_{.}\}$?

Fairly simply – we can do this by using a common index value for each season in the survival PIM:



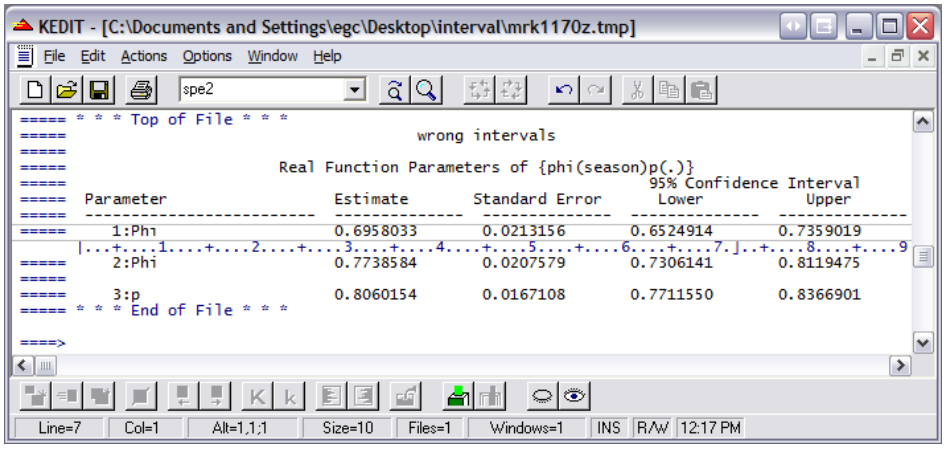
Here, the '1' index values correspond to the October → May season (recall that in this example, the first sampling occasion is assumed to be in October), and the '2' index values correspond to the May→ October season.

We see clearly (below) that model $\{\varphi_{(season)}p_{.}\}$ is not equivalent to model $\{\varphi_{.}p_{.}\}$:



Model	AICc	Delta AICc	AICc Weight	Model Likelihood	No. Par.	Deviance
{phi(season)p(.)}	2129.4722	0.0000	0.86816	1.0000	3	96.1489
{phi(.)p(.)}	2133.2418	3.7696	0.13184	0.1519	2	101.9285

We see from the parameter estimates for model $\{\varphi_{(season)}p_{.}\}$ (shown below) that the values for each season are very close to what we expected: 0.6958 is very close to $0.95^7 = 0.6983$, and 0.7739 is also very close to $0.95^5 = 0.7738$.



OK, all is well, right? Well, not quite. Suppose you wanted to test the hypothesis that monthly survival is the same between seasons. How would you do this? Well, you could derive an estimate of monthly survival from each of these seasonal estimates by taking the appropriate root of the estimated value. For example, for October → May, which is a 7 month interval, the estimated monthly survival probability is $\sqrt[7]{0.6958} = 0.9495$, and for May → October, which is a 5 month interval, the estimated survival probability is $\sqrt[5]{0.7739} = 0.9500$. While it is clear that both of these estimates are virtually identical in this instance, in practice you would need to derive SE's for these values, and use a formal statistical test to compare them (deriving the SE's for the *n*th roots - or any other function - of various parameter estimates involves use of the *Delta method* - see Appendix B).

How can we avoid these 'hand calculations'? Can we get **MARK** to give us the monthly estimates directly? In fact, we can, by correctly specifying the time intervals. Obviously we do so by entering the appropriate intervals once we've specified the appropriate number of sampling occasions. The key, however, is in deciding what is the *appropriate* interval. Suppose we're really interested in the monthly survival value, and whether or not these values differ between seasons. How can we test this hypothesis in **MARK**, if the number of months in the two seasons differs?

In fact, it is quite straightforward*, but first you need to know something about how **MARK** handles time intervals. Consider the following example – suppose that 3 consecutive years of live trapping are conducted (with the first year capturing only new or unmarked animals), then a year is missed, then 3 more consecutive years are conducted. Then, the time intervals for these 5 encounter occasions would be specified as

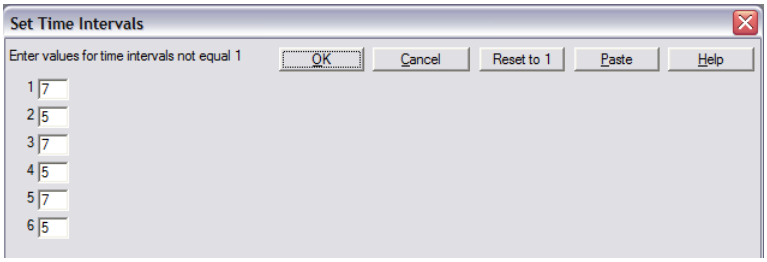
```
1 1 2 1 1
```

where the '2' indicates that the length of the time interval separating these 2 capture occasions is 2 years instead of 1 year like the other 4 intervals. The purpose of specifying the time intervals is to make the survival probabilities for each of the intervals comparable. Thus, the survival probability for all 5 of the above intervals will be an annual or 1 year probability, so that all can be constrained to be the same, even though the length of the time intervals to which they apply are not the same. The time interval is used as an *exponent* of the estimated survival probability to correct for the length of the time interval.

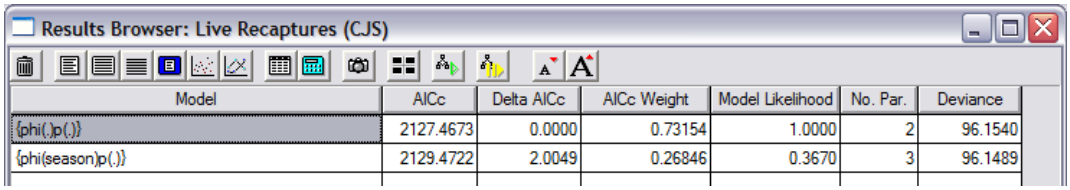
* At least, it is straightforward for simple live encounter models. Handling uneven intervals gets more complicated when we consider models where individuals 'move' between discrete states - these models are covered in later chapters.

To explain in more detail, unequal time intervals between encounter occasions are handled by taking the length of the time interval as the exponent of the survival estimate for the interval, i.e., $S_i^{L_i}$. For the typical case of equal time intervals, all unity (1), this function has no effect (since raising anything to the power of 1 has no effect). However, suppose the second time interval is 2 increments in length, with the rest 1 increment. This function has the desired consequences: the survival estimates for each interval are comparable, but the increased span of time for the second interval is accommodated. Thus, models where the same survival probability applies to multiple intervals can be evaluated, even though survival intervals are of different length. Moreover, you can use the exponent to derive estimates of survival for whatever interval you deem appropriate.

OK, back to our example - we're interested in monthly survival probabilities. To derive monthly survival probabilities, all you need to do is re-do the analysis, and enter the appropriate number of months:



Go ahead and re-run the analysis using the same two models. This time, however, we see that model $\{\varphi_{.}p_{.}\}$ is much better supported by the data than a model allowing for season differences:



Model	AICc	Delta AICc	AICc Weight	Model Likelihood	No. Par.	Deviance
{phi(.)p(.)}	2127.4673	0.0000	0.73154	1.0000	2	96.1540
{phi(season)p(.)}	2129.4722	2.0049	0.26846	0.3670	3	96.1489

Moreover, the estimated survival probability from this model (0.9497) is very close to the estimated true monthly survival probability used to simulate the data.

As a final test, suppose that instead of monthly estimates, you were interested in estimates calculated over 6 month intervals. You could derive 6-month (i.e., half-year) survival estimates (and corresponding standard errors) by hand, but can you use **MARK** to do this for you directly? Sure - all you need to do is re-scale both seasonal intervals in terms of the desired season length. How? Simply by using the fact that a 7 month interval is in fact (7/6) = 1.16$\dot{6}$ times as long as a 6 month interval, and that a 5 month interval is (5/6) = 0.83$\dot{3}$ times as long as a 6 month interval. So, all you need to do is enter these re-scaled intervals into **MARK**. Note however that the interval input window in **MARK** does not expand 'visibly' to handle non-integer intervals (or even integer intervals that are very large). This is not a problem, however. Simply go ahead and enter the values (we'll use 1.167 and 0.833, respectively, so: `1.167 0.833 1.167 0.833 1.167 0.833`).

Since the true monthly survival probability is 0.95, then if the season lengths were actually the same (6 months), then we would expect the estimated seasonal survival probabilities for both re-scaled seasons to be the same, $0.95^6 = 0.7351$, and that model $\{\varphi_{.}p_{.}\}$ should be much better supported than competing model $\{\varphi_{(season)}p_{.}\}$. In fact this is exactly what we see - the estimated 6-month survival probability from model $\{\varphi_{.}p_{.}\}$ is 0.7338, which is very close to the expected value of 0.7351.

end sidebar

OK - so we've considered some of the basics of building some models in **MARK**. But, what model, or models, should we make inference from? How do we establish whether or not some factor 'significantly' influences survival, or some other parameter of interest? What parameter estimates are most appropriate to report? Of course, these are in fact *the* critical questions underlying the exercise of fitting models to data in the first place. We begin addressing them in the next section.

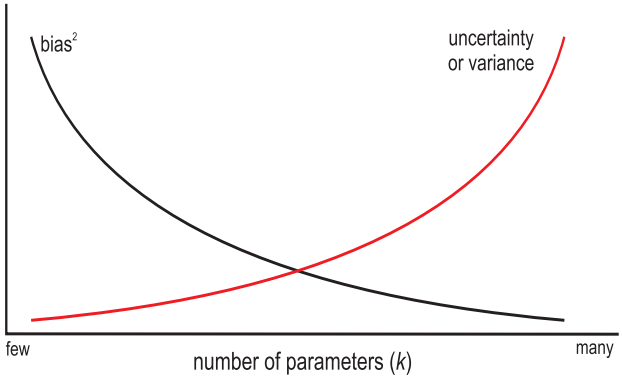
4.3. Model selection - the basics

In simplest terms, we might express our objective as trying to determine the best model from the set of approximating models we've fit to the data. How would we identify such a 'best model'? An intuitive answer would be to select the model that 'fits the data the best' (based on some statistical criterion). However, there is a problem with this approach - the more parameters you put into the model, the better the fit (analogous to ever-increasing R^2 in a multiple regression as you add more and more terms to the model). As such, if you use a simple measure of 'fit' as the criterion for selecting a 'best' model, you'll invariably pick the one with the most parameters. So, for our analysis of the Dipper data we would select model $\{\varphi_t p_t\}$ as our 'best' model, simply because it has the lowest deviance (36.4013), which of course it must since it has more parameters (11) than the other 3 models in the model set:

Results Browser: Live Recaptures (CJS)

Model	AICc	Delta AICc	AICc Weight	Model Likelihood	No. Par.	Deviance
{phi(.)p(.)}	322.5527	0.0000	0.96003	1.0000	2	41.8147
{phi(t)p(.)}	330.0567	7.5040	0.02253	0.0235	7	38.8147
{phi(.)p(t)}	330.6794	8.1267	0.01650	0.0172	7	39.4374
{phi(t)p(t)}	336.4343	13.8816	0.00093	0.0010	11	36.4013

Great, right? Don't we want to maximize the fit of our models to the data? Well - it's not quite that simple. While adding more parameters increases the *fit* of the model to the data, you pay a price in so doing - that price is parameter *uncertainty* (or variance).



This figure represents the trade-off between squared bias and variance versus the number of estimable parameters in the model. With an increasing number of parameters, the squared bias of the estimates of the individual parameters goes down. But. . .greater parameter uncertainty. How can we find a good, defensible compromise between the two? Treating the issue as a simple problem of 'balancing' variance and bias, one approach is to make use of something called the AIC.

4.3.1. The AIC, in brief...

The AIC (which in fact is an acronym for 'another information criterion', but is almost universally 'read' as 'Akaike's Information Criterion', after Hirotugu Akaike, who first described it in 1973*) comes to us from the world of information theory.

How does the AIC achieve an optimal 'balance' between precision and fit? While the 'deep theory' is somewhat dense (translation: not entirely trivial), in purely mechanical terms, it's pretty easy to see how the AIC works. The AIC is calculated for a particular model as

$$\mathrm{AIC} = -2\ln\mathcal{L}(\hat{\boldsymbol{\theta}} \mid \mathrm{data}) + 2K$$

where $\mathcal{L}$ is the *model likelihood* ($\boldsymbol{\theta}$ represents the vector of the various parameter estimates given the data), and K is the number of parameters in the model.

In general, the fit of the model to the data is 'indicated' by the model likelihood (maximum likelihood estimation was introduced in Chapter 1). Thus, as the fit of the model goes up, the likelihood of the model (given the data) goes up (and thus $-2\ln\mathcal{L}$ goes down). However, as indicated in the preceding figure, the greater the number of parameters, the greater the parameter uncertainty or variance. Thus, as the fit of the model increases, $-2\ln\mathcal{L}$ goes down – and for a given number of parameters, the AIC declines. Conversely, for a given fit, if it is achieved with fewer parameters (lower K), then the calculated AIC is lower. The $2K$ term, then, is the *penalty* for the number of parameters. As K goes up, likelihood goes down, but this is balanced by the penalty of adding the term $2K$.

So, one strictly utilitarian interpretation of the AIC is that the model with the lowest AIC is the 'best' model because it is most parsimonious given the data - best fit with fewest parameters. However, more formally, and perhaps more importantly at least conceptually, the model with the lowest AIC within the candidate set of approximating models can be shown to be the model which is closest to 'full truth' - which is not known (and is not contained in the candidate model set).

Say, what? The next couple of pages are somewhat tough going, but we recommend you at least try to read through the following to get a better 'feel' for the derivation – and application – of the AIC in model selection. Start by imagining a model f which represents full truth. Such a model might exist in theory, but we will never be able to fully specify it. Consider an approximating model g. We use the term *approximating* for g since g (and in fact any model) is an approximation of truth. Our goal in model selection is (ultimately) to determine which of our models minimizes the difference (distance) between g and f. In the 1950's, Kullback and Leibler determined that if $I(f,g)$ represents the 'information' lost when model g is used to approximate full truth f, then $I(f,g)$, the distance between a model g and full truth f, is given as

$$I(f,g) = \int f(x)\ln\left(\frac{f(x)}{g(x \mid \boldsymbol{\theta})}\right)dx,$$

Here f and g are probability distributions. The verbal description of $I(f,g)$ is that it represents the distance from model g to model f. Alternatively, it is the information lost when using g to approximate f.† As above, $\boldsymbol{\theta}$ represents the vector of the various parameters used in the specification of g.

It might be helpful to consider the form of Kullback-Leibler (K-L) information for discrete probability models (since they are somewhat easier to grasp). Let the true state of the system be

$$f = \{p_1, p_2, \ldots, p_k\}$$

* Akaike, Hirotogu, 1973. Information Theory and an Extension of the Maximum Likelihood Principle. In: B. N. Petrov and F. Csaki, eds. *Second International Symposium on Information Theory*. Budapest: Akademiai Kiado, pp. 267-281.)

† The negative of K-L information is Boltzmann's entropy, $H = -I(f,g)$, a fundamental concept in statistical thermodynamics.

Here there are k possible outcomes of the underlying random variable – the true probability of the ith outcome is given by p_i. Let the model *approximating* the state be

$$g = \{\gamma_1, \gamma_2, \ldots, \gamma_k\}$$

where γ_i represents the approximating probability distribution for the ith outcome. (Note that in the discrete case, $0 < p_i < 1, 0 < \gamma_i < 1$, and $\sum p_i = \sum \gamma_i = 1$).

Thus, the K-L information between models f and g is defined for discrete distributions to be

$$I(f, g) = \sum_{i=1}^{k} p_i \ln\left(\frac{p_i}{\gamma_i}\right)$$

After log-transforming, we write

$$I(f, g) = \sum_{i=1}^{k} p_i \ln p_i - \sum_{i=1}^{k} p_i \ln \gamma_i$$

(As an aside, you may recognize the first of the two terms in this difference as H, the Shannon-Weiner diversity index, another information-based measure.)

Now, back to the more general form – the integral form for $I(f, g)$ can be written equivalently as a difference of integrals

$$\begin{aligned} I(f, g) &= \int f(x) \ln f(x) dx - \int f(x) \ln g(x \mid \boldsymbol{\theta}) dx \\ &= E_f[\ln f(x)] - E_f[\ln g(x \mid \boldsymbol{\theta})] \end{aligned}$$

where in the second line we make use of the fact that the form of each integral is that of an *expectation*.

$I(f, g)$ is known as the *Kullback-Leibler* (K-L) information, or distance. With a bit of thought, it is clear that a 'good' model is one where the distance between the model g and 'truth' f is as small as possible. In other words, a model which minimizes the K-L distance. But, if we don't (and can't ever) know truth, then how can we 'estimate' the K-L distance for a given model? In fact, we can't, but it turns out that doesn't matter. We can make use of *relative* K-L information instead.

What do we mean by 'relative' K-L information? Look at the RHS of the preceding equation:

$$I(f, g) = E_f[\ln f(x)] - E_f[\ln g(x \mid \boldsymbol{\theta})]$$

The first expectation $E_f[\ln(f(x))]$ is 'truth', which clearly must be a constant across models. Thus,

$$\begin{aligned} I(f, g) &= \text{Constant} - E_f[\ln g(x \mid \boldsymbol{\theta})] \\ I(f, g) - \text{Constant} &= -E_f[\ln g(x \mid \boldsymbol{\theta})] \end{aligned}$$

The term $[I(f, g) - \text{Constant}]$ is called relative Kullback-Leibler information (or distance), and is the relative distance between truth (f) and the approximating model (g). Relative K-L information is measured on an interval scale, a scale without an absolute zero. (In fact, the absolute zero is truth and is no longer part of the expression.)

Why is the relative K-L information of interest? Suppose that in addition to our approximating model g we have a second approximating model h for the true state of nature f. The information lost in using

h to approximate f is given by the following

$$\begin{aligned} I(f,h) &= \int f(x)\ln f(x)dx - \int f(x)\ln h(x \mid \boldsymbol{\theta})dx \\ &= E_f[\ln f(x)] - E_f[\ln h(x \mid \boldsymbol{\theta})] \end{aligned}$$

Observe that $E_f[\ln f(x)]$ is a common term in the expression for both model g and model h. Thus, if we want to compare model g to model h it makes sense to consider the difference $I(f,g) - I(f,h)$. If we do so then we find

$$\begin{aligned} I(f,g) - I(f,h) &= (E_f[\ln f(x)] - E_f[\ln g(x \mid \boldsymbol{\theta})]) - (E_f[\ln f(x)] - E_f[\ln h(x \mid \boldsymbol{\theta})]) \\ &= E_f[\ln h(x \mid \boldsymbol{\theta}] - E_f[\ln g(x \mid \boldsymbol{\theta})] \end{aligned}$$

OK, but where does that leave us? Look closely – notice that in the preceding bit of algebra, $E_f[\ln f(x)]$ has canceled out. So what? Again, recall that $E_f[\ln f(x)]$ represents truth! So, if our goal is to compare two models, truth drops out the comparison – which is a good thing since we cannot ever know what truth is. Its absolute magnitude has no meaning. It is only useful for measuring how far apart two approximating models are. This last expression represents the difference in *relative* Kullback-Leibler information between two models. So if our only goal is model comparison then our objective can be more limited. Rather than estimate K-L information we can estimate instead *relative* K-L information - the information lost when model g_i is used to approximate full reality (f). Another view of this is the *distance* between model g_i and full reality.

In either case, it seems compelling that one would want to select the model in the set of R models ($g_1, g_2, \ldots, g_R$) that minimizes K-L information loss. That is, we want the model from within the model set that loses the least information about full reality, hence, the model that is closest to full reality in the current model set (Fig. 4.1)*.

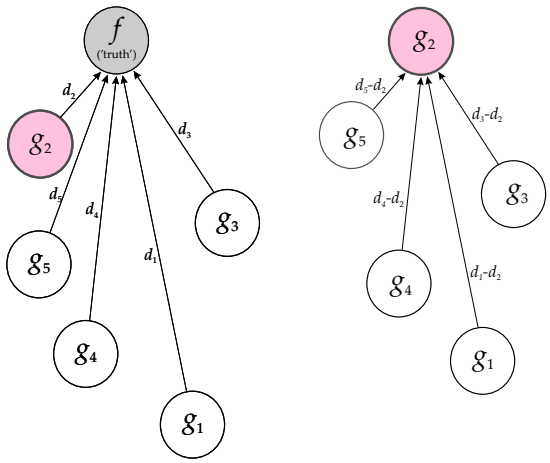


Figure 4.1: *Kullback-Leibler information is shown (at left) as the distances (d_i) between full reality (f) and the various models (g_i). The Δ values (right) provide the estimated distance of the various models to the best model (in this case, model g_2). These values are on the scale of information irrespective of the scale of measurement or type of data.*

* Burnham, Anderson & Huyvaert (2011) - *Behavioural Ecology & Sociobiology*, **65**, 23-35.

While all might seem well, recall that in our approximating model we typically won't know the exact value of $\boldsymbol{\theta}$. Instead we will have to use an estimate, $\hat{\boldsymbol{\theta}}$. To account for this additional uncertainty, Akaike suggested that what we should do is to calculate the *average* value of relative K-L information over all possible values of $\boldsymbol{\theta}$. In terms of expectation we would call this quantity *expected* relative K-L information and write it as

$$E\left[E_f\left[\ln g(x \mid \boldsymbol{\theta})\right]\right]$$

Akaike showed that an asymptotically unbiased estimator of the relative expected K-L distance from 'truth' could be calculated as

$$\ln \mathcal{L}(\hat{\boldsymbol{\theta}} \mid \text{data}) - K$$

where $\mathcal{L}(\hat{\boldsymbol{\theta}} \mid \text{data})$ is the log likelihood function for an approximating model evaluated at the maximum likelihood estimate of the parameter set $\boldsymbol{\theta}$, and where K is the number of parameters estimated in maximizing the likelihood of the model. Akaike then defined 'an information criterion' (AIC), by multiplying by -2 (for 'historical reasons', it seems) to get the familiar

$$\text{AIC} = -2 \ln \mathcal{L}(\hat{\boldsymbol{\theta}} \mid \text{data}) + 2K$$

Thus, as suggested earlier, one should select the model that yields the smallest value of AIC among the models in the candidate model set, not simply because it provides some 'balance' between precision and fit, but because this model is estimated to be the 'closest' to the unknown reality that generated the sample data, from among the candidate approximating models being considered. In other words, you should use the AIC to select the fitted approximating model that is estimated to be closest to the unknown truth (i.e., which minimizes the K-L distance). This, of course, amounts to selecting the model with the lowest AIC, among those models in the candidate model set. We emphasize here that the theory guarantees that the model with the lowest AIC has the smallest K-L distance amongst the models in the model set, conditional on the model set being specified *a priori*.

Returning to the Dipper analysis, we note that even though model $\{\varphi_t p_t\}$ has the lowest deviance (best fit; 36.40), it also has the greatest number of parameters (11) and the highest AIC value. In contrast, the model deviance for model $\{\varphi_{\cdot} p_{\cdot}\}$ is the greatest (fits the least well), but because it uses only 2 estimated parameters, it in fact has the lowest AIC of the 4 models.

4.3.2. Some important refinements to the AIC

While Akaike derived an asymptotically unbiased estimator of K-L information, the AIC may perform poorly if there are too many parameters in relation to the size of the sample. A small-sample (second order) bias adjustment led to a criterion that is called AIC_c (Sugiura 1978; Hurvich & Tsai 1989), that accounts for differences in effective sample size.

$$\text{AIC}_c = -2 \ln \mathcal{L}(\hat{\boldsymbol{\theta}}) + 2K\left(\frac{n}{n - K - 1}\right)$$

where the penalty term ($2K$) is multiplied by the correction factor $n/(n - K - 1)$.

This expression can be rewritten equivalently as

$$\text{AIC}_c = -2 \ln \mathcal{L}(\hat{\boldsymbol{\theta}}) + 2K + \left(\frac{2K(K+1)}{n - K - 1}\right)$$

where n is sample size. Because AIC and AIC_c converge when sample size is large, one can always

use AIC_c. As such, the AIC values reported by **MARK** are by default based on this modified (correct) version of the AIC.

We'll talk about additional modifications to the AIC, particularly to account for lack of fit (c), in the next chapter, but for the moment, conceptually at least, the AIC is simply the sum of 2 times the negative log of the model likelihood and 2 times the number of parameters, adjusted for sample size.

begin sidebar

Maximum likelihood, least-squares, and AIC

For those of you with a limited background in statistics – say a decent course in analysis of variance, or regression – you may at this point be wondering what the connection is between what you learned in those classes (which are often based on 'sums of squares' and 'residual sums of squares' and MLE, and AIC).

We'll show the connections by means of a fairly familiar example - the MLE for the mean, variance and standard deviation of the normal distribution. From any decent statistics text, the *pdf* (probability distribution function) for the normal distribution is

$$f(x) = \frac{1}{\sigma_x\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\bar{x}}{\sigma_x}\right)^2}$$

from which the likelihood is given as

$$\mathcal{L}\left(x_1, x_2, \ldots, x_N | \bar{x}, \sigma_x\right) = \mathcal{L} = \prod_{i=1}^{N}\left(\frac{1}{\sigma_x\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x_i-\bar{x}}{\sigma_x}\right)^2}\right)$$
$$= \frac{1}{(\sigma_x\sqrt{2\pi})^N} e^{-\frac{1}{2}\sum_{i=1}^{N}\left(\frac{x_i-\bar{x}}{\sigma_x}\right)^2}$$

Then

$$\ln(\mathcal{L}) = -\frac{N}{2}\ln(2\pi) - N\ln\sigma_x - \frac{1}{2}\sum_{i=1}^{N}\left(\frac{x_i-\bar{x}}{\sigma_x}\right)^2$$

Taking the partial derivatives of $\mathcal{L}$ with respect to each one of the parameters and setting them equal to zero yields,

$$\frac{\partial\mathcal{L}}{\partial\bar{x}} = \frac{1}{\sigma_x^2}\sum_{i=1}^{N}(x_i-\bar{x}) = 0$$

and,

$$\frac{\partial\mathcal{L}}{\partial\sigma_x^2} = -\frac{N}{\sigma_x} + \frac{1}{\sigma_x^3}\sum_{i=1}^{N}(x_i-\bar{x})^2$$

Solving these two equations simultaneously for $\bar{x}$ and σ_x yields

$$\bar{x} = \frac{1}{N}\sum_{i=1}^{N}x_i \qquad \sigma_x^2 = \frac{1}{N}\sum_{i=1}^{N}(x_i-\bar{x})^2$$

Now, consider again the $\ln\mathcal{L}$ expression:

$$\ln(\mathcal{L}) = -\frac{N}{2}\ln(2\pi) - N\ln\sigma_x - \frac{1}{2}\sum_{i=1}^{N}\left(\frac{x_i-\bar{x}}{\sigma_x}\right)^2$$

You might (should) remember from your statistics class that the residual sums of squares (RSS) is given as

$$RSS = \left(x_i - \bar{x}\right)^2$$

Thus, we can rewrite the $\ln \mathcal{L}$ as

$$\begin{aligned}\ln(\mathcal{L}) &= -\frac{N}{2}\ln(2\pi) - N\ln\sigma_x - \frac{1}{2}\sum_{i=1}^{N}\left(\frac{x_i - \bar{x}}{\sigma_x}\right)^2 \\ &= -\frac{N}{2}\ln(2\pi) - N\ln\sigma_x - \frac{1}{2}\left(\frac{RSS}{\sigma_x^2}\right)\end{aligned}$$

We see clearly that minimizing the RSS is equivalent to minimizing the likelihood.

Finally, differentiating this expression with respect to σ^2 yields

$$\hat{\sigma^2} = \frac{RSS}{N}$$

which when substituted into the likelihood expression yields

$$\begin{aligned}\ln(\mathcal{L}) &= -\left(\frac{N}{2}\right)\ln(2\pi) - N\ln\sigma_x - \frac{1}{2}\left(\frac{RSS}{\sigma_x^2}\right) \\ &= C - \frac{N}{2}\ln\left(\frac{RSS}{N}\right) - \frac{N}{2}\end{aligned}$$

where C is the constant $-(N/2)\ln(2\pi)$.

Thus, we can write the AIC in terms of RSS as

$$\begin{aligned}\text{AIC} &= -2\ln\mathcal{L} + 2K \\ &= N\ln\left(\frac{RSS}{N}\right) + 2K\end{aligned}$$

end sidebar

4.3.3. BIC - an alternative to the AIC

While the AIC has been shown to be a good omnibus approach to model selection, there are some theoretical considerations which may justify consideration of an alternative model selection criterion. One such measure is the BIC (Bayes Information Criterion), which can be used instead of the AIC in **MARK** - simply select '**File | Preferences | Display BIC instead of AIC**'. BIC or QBIC are alternative model selection metrics to AIC_c or QAIC_c. The number of parameters in the model is K.

The BIC depends on the number of parameters as

$$\text{BIC} = -2\ln\mathcal{L}(\hat{\boldsymbol{\theta}}) + K\ln n_e$$

and as does the QBIC (*quasi*-BIC)

$$\text{QBIC} = \frac{-2\ln\mathcal{L}(\hat{\boldsymbol{\theta}})}{\hat{c}} + K\ln n_e$$

where n_e is the effective sample size, and $\hat{c}$ is an adjustment for lack of fit of the general model to the

data (this is introduced in the next chapter). If you select the BIC, model weights and model likelihood are also computed using BIC instead of AIC_c, so that model averaging is also conducted from the BIC.

When should you use BIC versus AIC? This is a very deep question, and we can only briefly describe some of the issues here. Much of the following is abstracted from the following paper:

Burnham, K.P. & D.R. Anderson. (2004) Multimodel inference - understanding AIC and BIC in model selection. *Sociological Methods & Research*, **33**, 261-304.

In general, recent research (much of it collated in the Burnham & Anderson paper) suggests there are distinct contexts (say, model sets consisting of simple versus complex models) for which BIC outperforms AIC (generally, when the approximating models in the model set are simple - relatively few 'main effect' factors), or where AIC outperforms BIC (when models are multi-factorial, and generally more complex). AIC is often claimed (equally often without much empirical support) to 'over-fit' - select models which are overly parameterized (relative to the true generating model), whereas the BIC has been suggested to 'under-fit' - select models which are less parameterized than the true generating model.*

Why? While the technical reasons for any difference in 'relative performance' are complex, there is a simple intuitive argument based on the fundamental difference in how the AIC and BIC are estimated. Consider the differences in the following two equations:

$$\mathrm{AIC} = -2\ln\mathcal{L}(\hat{\boldsymbol{\theta}}) + 2K$$

$$\mathrm{BIC} = -2\ln\mathcal{L}(\hat{\boldsymbol{\theta}}) + K\ln n_e$$

So, in simplest terms, the difference between the AIC and the BIC is in terms of the multiplier for K in the 'penalty term' for the number of parameters: 2 for the AIC, versus $\ln n_e$ for the BIC. Clearly, $2 \neq \ln n_e$. But, more importantly, the multiplier for the AIC (2) is a constant scalar, whereas for the BIC it scales as a function of the effective sample size. Recall that the larger the penalty, the simpler the selected model (all other things being equal). As a result, AIC tends to perform well for 'complex' true models and less well for 'simple' true models, while BIC does just the opposite. In practice the nature of the true model, 'simple' or 'complex', is never known. Thus a data driven choice of model complexity penalty would be desirable. This is an active area of research. It is important to remember that the AIC is an estimate of the expected Kullback-Leibler discrepancy (discussed earlier), while BIC is (in fact) an asymptotic Bayes factor (see paper by Link & Barker (2006) Model weights and the foundations of multimodel inference. *Ecology*, **87**, 2626-2635). Since each method was derived with different motivations, it is not surprising that they have quite different theoretical properties.

While a full discussion of these issues is beyond the scope of what we want to present here, it is important to note that focus should not be on 'which model selection criterion is best?', but remembering that 'model selection should be considered as the process of making inference from a set of models, not just a search for a single best model'. As such, whenever possible, use model averaging. Not only does this account for model selection uncertainty regarding estimated parameters and weight of evidence for each approximating model, but also, differences between inference under AIC_c versus BIC diminish under model averaging.

Note: why doesn't **MARK** allow you to show both the AIC and BIC values/weights in the same browser? Simple - to help discourage you from using a side-by-side comparison of the two to guide your model selection - doing so would amount to little more than a different form of *post hoc* data dredging. Of course you could do it yourself 'by hand' - but don't let us catch you!

* 'All generalizations are false, including this one...' - Alexandre Dumas, or Mark Twain, depending on your source.

4.4. Using the AIC for model selection - simple mechanics...

The basic mechanics of using AIC_c for model selection in **MARK** are straightforward. The AIC_c is computed for each of the models in the candidate set, and the models are automatically sorted in descending order based on the AIC_c (i.e., the most parsimonious model – the one with the smallest AIC_c value – is placed at the top of the results).

OK - so you run **MARK**, and calculate the AIC_c for each model. What do you do if, say, the model with the lowest AIC_c differs from the next-lowest by only a small amount? How much 'support' is there for selecting one model over the other? Note - we intentionally use the word *support*, rather than *statistical significance*. We'll deal with the issue of 'significance', and related topics, shortly.

As a first step, the models should be calibrated to provide an index of 'relative plausibility' (i.e., the likelihood of the model given the model set), using what are known as *normalized Akaike weights*. These weights (w_i) are calculated for each approximating model (i) in the candidate model set as

$$w_i = \frac{\exp\left(\frac{-\Delta\text{AIC}}{2}\right)}{\sum\left\{\exp\left(\frac{-\Delta\text{AIC}}{2}\right)\right\}}$$

What are we doing here, and why? To help understand the basic idea behind normalized AIC weights, and how they are calculated, consider the concept of the likelihood of the *parameters* $\boldsymbol{\theta}$ given a model g_i, and some data x

$$\mathcal{L}(\hat{\boldsymbol{\theta}} \mid x, g_i)$$

We can extend this basic idea to the concept of the likelihood of the *model* given the data

$$\mathcal{L}(g_i \mid x) \propto e^{-\frac{1}{2}\Delta_i}$$

where Δ_i is the difference in the AIC value between the model i and the model with the lowest AIC. Note that the $-1/2$ term simply cancels out the fact that Akaike multiplied through by -2 to define his AIC. So, the likelihood of a model, given the data, is proportional to the difference in AIC between that model, and the model in the model set with the lowest AIC. Normalizing them creates a set of positive values that sum to one (which lends to the interpretation of relative or proportional support in the data for a given model, among the models in the candidate model set).

OK, fine, but you might be asking 'why is the likelihood of a model, given the data, proportional to the difference in AIC between that model and the candidate model with the lowest AIC?'. The following might help. We note that for model i, $\Delta\text{AIC}=\text{AIC}_i\text{-AIC}_0$, where AIC_0 is the minimum AIC in the candidate model set (i.e., the most parsimonious model).

Given that $\text{AIC}=-2\ln\mathcal{L}+2K$, then

$$\begin{aligned}\Delta\text{AIC} &= \text{AIC}_i - \text{AIC}_0 \\ &= (-2\ln\mathcal{L}_i + 2K_i) - (-2\ln\mathcal{L}_0 + 2K_0) \\ -\frac{\Delta\text{AIC}}{2} &= \ln\mathcal{L}_i - K_i - \ln\mathcal{L}_0 + K_0\end{aligned}$$

Thus,

$$\exp\left(\frac{-\Delta AIC_i}{2}\right) = \left(\frac{\mathcal{L}_i}{\mathcal{L}_0}\right)\exp(K_0 - K_i)$$

So, the term $\exp(-\Delta_i/2)$ in the expression used to calculate AIC weights is in fact a *likelihood ratio*, corrected for the difference in the number of parameters estimated from the models.

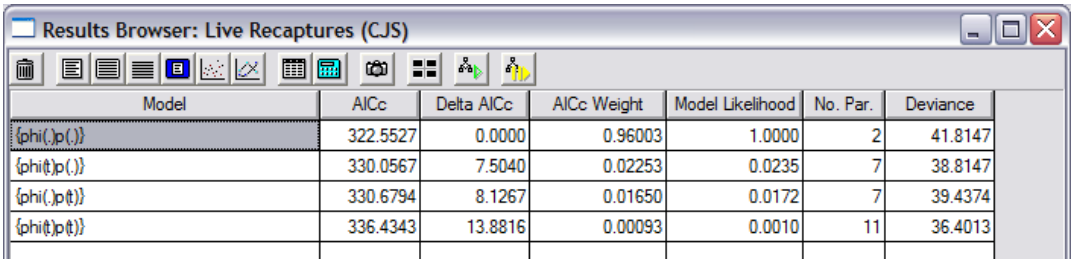
How are these weights used? A given w_i is considered as the weight of evidence in favor of model i as being the actual K-L best model in the set. These are termed *model probabilities* (in fact, they are also formally Bayesian posterior model probabilities; Burnham & Anderson 2004). So, w_i is the probability that model i is the actual K-L best model in the set. The bigger the w_i value, the bigger the weight. The bigger a Δ_i value is, the less plausible is model i as being the actual K-L best model of full reality, based on the design and sample size used.

For example, consider the following set of models, with their ΔAIC_c values and Akaike weights.

Model	ΔAIC	Akaike weight (w_i)
1	1.6	0.278
2	0.0	0.619
3	7.0	0.084
4	13.5	0.001
5	4.0	0.084
total		1.000

Here, model 2 is clearly the best (largest AIC weight), but how much better is it than the next best model (model 1)? The Akaike weights let us state that the best model (model 2) is over twice as well supported as the next best model (model 1), since (0.619/0.278) = 2.23. The remaining models (3, 4 and 5) have essentially no support in the data.

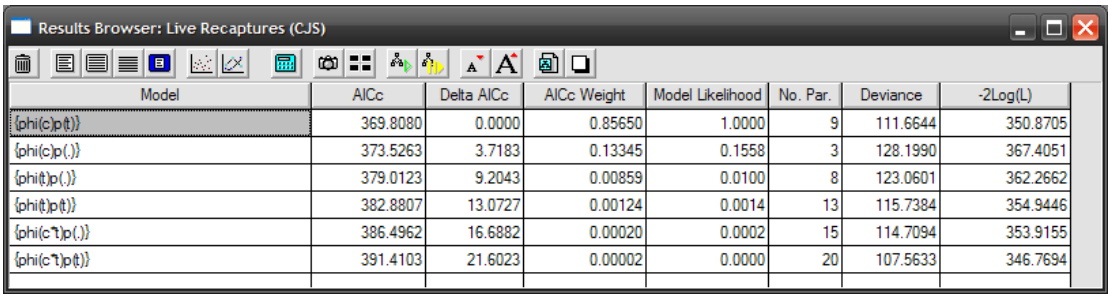
MARK calculates Akaike weights automatically (if the are not already in the results browser, they can be 'turned on' under '**`File | Preferences`**'). For our Dipper analysis, here again are the AIC values, the ΔAIC values, and their relative (normalized) weights.



Model	AICc	Delta AICc	AICc Weight	Model Likelihood	No. Par.	Deviance
{phi(.)p(.)}	322.5527	0.0000	0.96003	1.0000	2	41.8147
{phi(t)p(.)}	330.0567	7.5040	0.02253	0.0235	7	38.8147
{phi(.)p(t)}	330.6794	8.1267	0.01650	0.0172	7	39.4374
{phi(t)p(t)}	336.4343	13.8816	0.00093	0.0010	11	36.4013

In this case, the results are clear - model $\{\varphi_{.}p_{.}\}$ is much better supported than any other model - the AIC for the next best model differs by 7.50, and has approximately 43-times less support than the best model.

Consider again are the results from the analysis of the swift data (shown at the top of the next page). Again, the results are quite clear - model $\{\varphi_c p_t\}$ is much better supported by the data than any other model in the candidate model set. The AIC_c for the next best model (model $\{\varphi_c p_{.}\}$) differs by 3.72, and has approximately 6-times less support than the best model. Note that in this example, unlike for the Dipper data, the model with the fewest parameters is not the most parsimonious model. Again, ranking based on the AIC balances fit and precision, given the data.


Results Browser: Live Recaptures (CJS)

Model	AICc	Delta AICc	AICc Weight	Model Likelihood	No. Par.	Deviance	-2Log(L)
{phi(c)p(t)}	369.8080	0.0000	0.85650	1.0000	9	111.6644	350.8705
{phi(c)p(.)}	373.5263	3.7183	0.13345	0.1558	3	128.1990	367.4051
{phi(t)p(.)}	379.0123	9.2043	0.00859	0.0100	8	123.0601	362.2662
{phi(t)p(t)}	382.8807	13.0727	0.00124	0.0014	13	115.7384	354.9446
{phi(c*t)p(.)}	386.4962	16.6882	0.00020	0.0002	15	114.7094	353.9155
{phi(c*t)p(t)}	391.4103	21.6023	0.00002	0.0000	20	107.5633	346.7694

In the preceding, you may have noticed a column in the results browser labeled '**model likelihood**'. In fact, this column does not represent the ML of the model (and is therefore not the value used to calculate the AIC), but rather is the AIC (or BIC) *weight* for the model of interest divided by the AIC (or BIC) weight of the best model in the browser. For example, the model likelihood for model $\{\varphi_c p_.\}$ is (0.13345/0.85650)=0.1558. This likelihood value is the strength of evidence of this model relative to other models in the set of models considered, and is the reciprocal of the evidence ratio (discussed in the - `sidebar` -, below). You can add a column to the browser showing the maximized likelihood for the model (i.e., $-2\ln\mathcal{L}$) by selecting that option in '**File | Preferences**'.

4.4.1. AIC weights and 'rules-of-thumb' for model selection

What do you do if, say, the model with the lowest AIC differs from the next-lowest by only a small amount? How much 'support' is there for selecting one model over the other? Note - we use the word *support*, rather than *statistical significance*. In general, Anderson & Burnham recommend several 'rules of thumb' to select among models, based on differences in AIC. When the difference in AIC between 2 models (ΔAIC) is < 2, then we are reasonably safe in saying that both models have approximately equal weight in the data. If $2 < \Delta\text{AIC} < 7$, then there is considerable support for a real difference between the models, and if $\Delta\text{AIC} > 7$, then there is strong evidence to support the conclusion of differences between the models. (*Note*: the motivation for these 'rules of thumb' is addressed in the following - `sidebar` -).

begin sidebar

AIC weights and evidence ratios

Evidence can be judged by the relative likelihood of model pairs, or equivalently the ratio of AIC weights w_i/w_j. Such ratios are termed *evidence ratios*, and represent the evidence about fitted models as to which is 'better' in the information theoretic sense.

In general, interest focuses on the ratio w_1/w_j, where model 1 is the estimated best model, and j indexes the rest of the models in the model set. Evidence ratios provide a measure of the *relative likelihood* of one hypothesis (model) versus another. Here, likelihood has a technical meaning, that can be quantified and should not be confused with probability. For example, if person A holds five raffle tickets and person B has one, person A is five times more *likely* to win than person B. We do not know the absolute *probability* of either person winning without knowing the total number of raffle tickets.

Consider the results from the swift example, shown at the top of this page. In this case, the evidence ratio gauges the relative support for 6 models in the candidate model set. Given the available data, a model where survival is fixed (i.e., constant) among years, but which differs between colonies, and where encounter probability varies over time, but not between colonies $\{\varphi_c p_t\}$ is (0.8565/0.1335) = 6.42 times more likely than a model where survival is again fixed (i.e., constant) among years, but which differs between colonies, and where encounter probability does not vary between colonies or over time $\{\varphi_c p_.\}$. This suggests little evidence for time variation in encounter probability.

As noted earlier, you can select an option in '**File | Preferences**', to add a column labeled '**Model**

Likelihood' to the results browser. What this column is, in fact, is the AIC_c weight for the model of interest divided by the AIC_c weight of the best model. In other words, this value is the strength of evidence of this model relative to other models in the set of models considered, and is precisely the inverse of the classical weight of evidence, as defined above. For example **MARK** reports that the relative likelihood of model $\{\varphi_c p_.\}$ to model $\{\varphi_c p_t\}$ is 0.1558. Note that $(1/0.1558) = 6.42$.

Evidence ratios are invariant to other models in the model set (whereas model weights depend on all the other models in the candidate model set). Inference should be about models and parameters, given data; however, we note that P-values are probability statements about *data*, given null models. Model probabilities and evidence ratios provide a means to make inference directly about models and their parameters, given data.

Evidence ratios are a continuous measure. And, as noted by Burnham & Anderson (2002), there is a marked nonlinearity in evidence ratios as a function of the ΔAIC_i values. If we consider the ratio

$$\frac{w_1}{w_j} \equiv \frac{1}{e^{-1/2\Delta_j}} \equiv e^{1/2\Delta_j}$$

as a comparison of the evidence for the best model (lowest AIC) compared with any other model j, then we can generate following table:

Δ_j	2	4	8	10	15
evidence ratio	2.7	7.4	54.6	148.4	1808.0
model likelihood	0.3704	0.1352	0.0183	0.0067	0.0006

It is just this nonlinearity in the relationship between ΔAIC and the evidence ratio which leads to the 'rule of thumb' introduced earlier. As suggested by Anderson & Burnham, when the difference in AIC between two models (ΔAIC) is < 2, then we are reasonably safe is saying that both models have approximately equal weight in the data. From the preceding table, we see clearly that the model likelihood is $\gg \alpha = 0.05$. Conversely, if $\Delta AIC > \simeq 7$, then there is strong evidence to support the conclusion of real differences between the models.

From a practical standpoint, when reporting model selection results (in a paper, or report), it is useful to report both AIC weights and either model likelihoods or evidence ratios (reporting both would be redundant).

end sidebar

However, while rules of thumb are convenient, they don't quantify the degree of uncertainty in our model selection. What do we mean by 'uncertainty', in the context of model selection? In any analysis, there is uncertainty in terms of which model is the 'best model'. In our swift analysis, for example, we determined which model is the most parsimonious, but how far from 'truth' is this model? The most parsimonious model is merely the model which has the greatest degree of support in the data. It is not 'truth' - it merely does somewhat better at explaining variation in the data than do other models (we add in passing that '*All models are wrong, some are useful*' - G. Box). There is 'uncertainty' in terms of which model is the 'best model'. How can we measure, or at least account for, this uncertainty?

4.5. Model uncertainty: an introduction to model averaging

In the analysis of the swift data set we considered earlier in this chapter, we compared the survival probability of birds as a function of the quality of their nesting colony ('good' versus 'poor'). We came to the conclusion that there was some support in the data for a colony effect (the 2 most parsimonious models both had a colony effect in survival, and the sum of their respective normalized AIC weights was 0.985, indicating $\approx 98.5\%$ of the support in the data are for these 2 models). If we look at the estimates

from the most parsimonious model in our model set (model $\varphi_c p_t$), we see that the estimate of survival for the good colony was 0.77 (SE 0.041), while the estimate for the poor colony was 0.58 (SE 0.082). Our previous analysis seems to support the contention that this was a meaningful difference between the two colonies.

However, suppose you are charged with drafting a conservation plan for this population, and want to condition your recommendations on the possibility of differences in survival between the 2 colonies. Perhaps you want to use the estimates of survival in some form of model, projecting the likely consequences of one or more proposed actions. While how you might do this is obviously beyond the scope of this book (since it has little to do with **MARK** directly), it does raise at least one issue which is worth noting at this point. What should we use as the estimates of survival?

The obvious answer would be to use the estimates from the most parsimonious model alone. However, taking this approach clearly ignores one salient fact - the estimates of sampling variance from a given model do not include model selection uncertainty - they are conditional only on the model used for the estimation. In other words, using the estimates from a single model in the candidate model set, even if it is the most parsimonious model in the set, ignores model uncertainty. For example, for the swift analysis, model $\{\varphi_c p_t\}$ has a AIC_c weight of 0.8565, while model $\{\varphi_c p_\cdot\}$ has a AIC_c weight of 0.1335. While model $\{\varphi_c p_t\}$ is clearly better supported, there is still uncertainty – there is at least some chance (approximately 13% chance) that in fact model $\{\varphi_c p_\cdot\}$ is the correct model, relative to the other models in the candidate model set.

Since there is uncertainty in which model is the correct model, we might consider accommodating this uncertainty in the estimates we report (or use subsequently in some model). This is where 'model averaging' comes in. The simplest way to think of what model averaging is all about is to recall the concept of 'weighted average' from your basic statistical training. What we want to do is take the estimates from our various models, and weight them by the relative support for that model in the data. More precisely, we calculate an average value for a parameter θ by averaging over all models in the candidate model set with common elements in the parameter structure, weighted by normalized AIC model weights (*sensu* Buckland *et al*., 1997, Burnham & Anderson 2004):

$$\text{avg}(\hat{\theta}) = \hat{\bar{\theta}} = \sum_{i=1}^{R} w_i \hat{\theta}_i$$

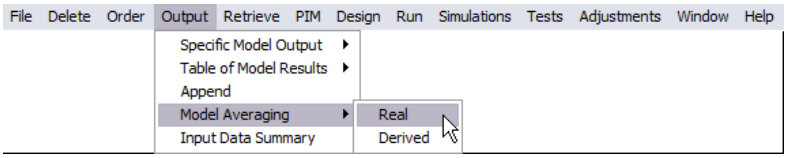
where w_i reflects the Akaike weight for model i. Hopefully, this makes some intuitive sense - we weight the estimates of the various parameters by the model weights.

Let's see how we actually do this in **MARK**. Suppose for example we're interested in reporting the live encounter probabilities for each year in the study. What would our 'best' estimates be for annual encounter probability? We see from the results browser that the most parsimonious model has time-dependence in p (i.e., p_t), while the next best supported model has constant p. If we were simply to report the estimates from the most parsimonious model, we would have:

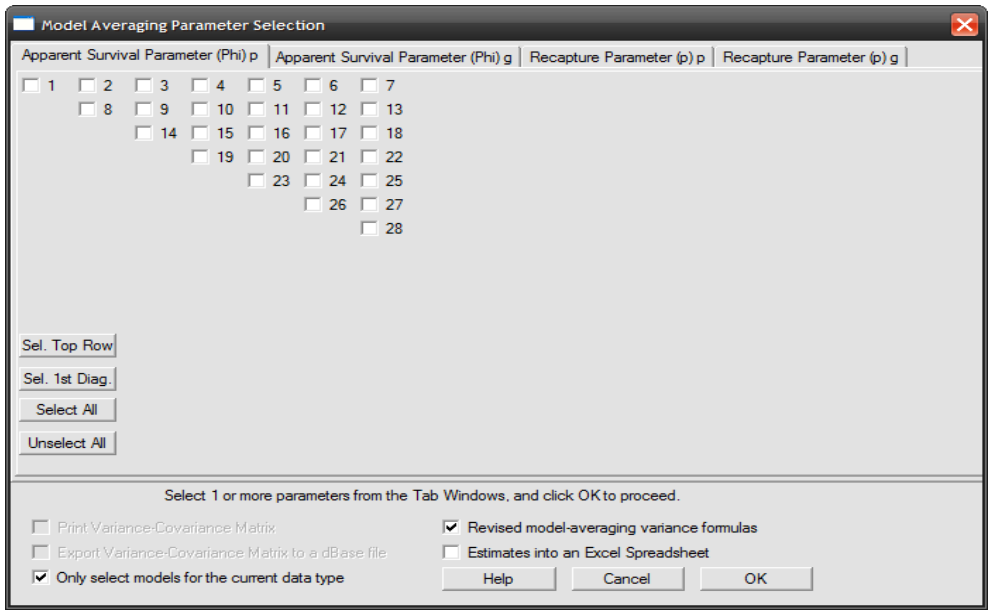
year	*estimate*	SE
2	0.909	0.086
3	0.729	0.103
4	0.537	0.114
5	0.698	0.104
6	0.858	0.088
7	0.860	0.105
8	0.463	0.095

Now, what would our 'model averaged values' be for these estimates?

To derive average values in **MARK**, pull down the '**Output**' menu, and select '**Model averaging**', and the '**Real parameters**'.



This will spawn the following window:

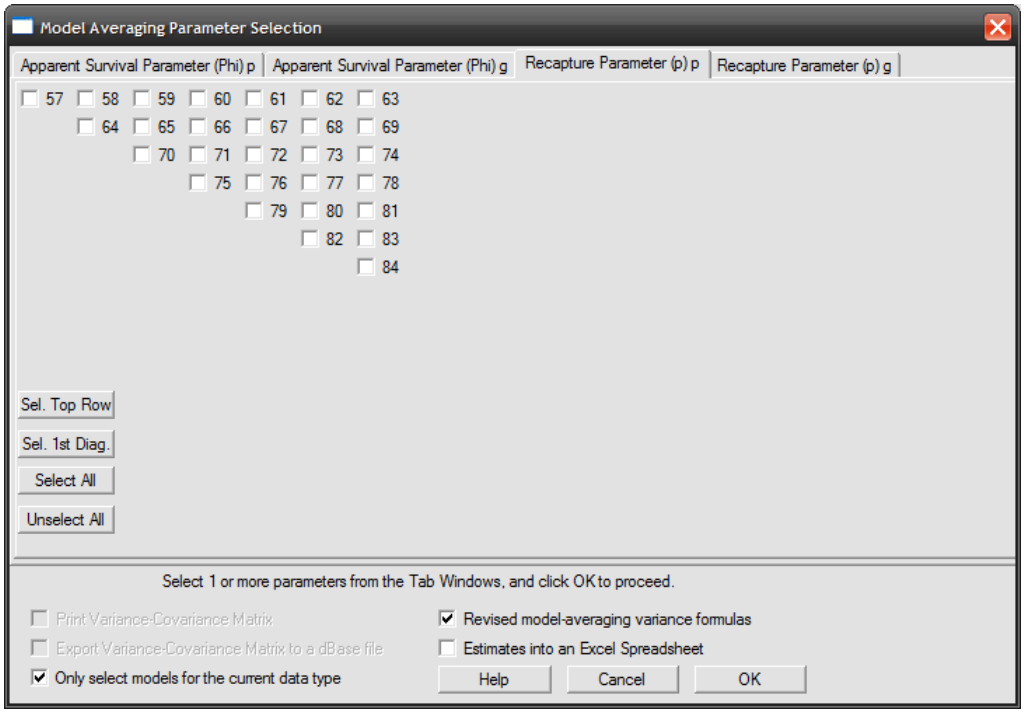


Notice along the top there are 4 'tabs', like the tabs on file folders. One tab for each of the 4 main parameters (survival for the poor colony, survival for the good colony, recapture for the poor colony, and recapture for the good colony). So far, so good. Notice also that there is a triangular matrix of 'check boxes' which is structurally equivalent to the structure of the PIMs. Structurally equivalent, but...what do the numbers represent? Clearly, they're not equivalent to the indexing we used in our analysis of the swift data set - are they?

No, as written, they're not, but...they're directly related. The numbering you see in the model averaging window corresponds to the index values that you would use in a PIM if the model had complete (`cohort` × `time`) dependence. Say...what?? Well, cohort models and other extensions of the simple time-dependent model is something we'll get to in chapter 7. But, for now, simply think of the indexing you see in the model averaging window as corresponding to the possibility that there is a different estimate for each time period (time-dependence), and that within each time-period, the estimate might vary as a function of what year the organism was marked and released (cohort-dependence). Since there are 28 combinations of time and cohort in our swift data set, that is why the model averaging window has 28 cells, numbered 1 to 28. The numbering is left to right within each cohort in succession. Again, do not

get too concerned at this point if the distinction between 'time period' and 'cohort' is a bit confusing – by the time you reach the end of chapter 7, you'll fully understand the distinction (we hope!).

First, we need to 'tell' **MARK** we want to average the recapture estimates across models. We'll start by selecting the '**`recapture parameter (p) P`**' tab, corresponding to the recapture probabilities for the poor (**P**) colony. Do this by clicking on that tab in the model averaging window. Now, you'll see the triangular matrix is numbered 57 through 84 (shown at the top of the next page). We're interested in annual estimates of p. As such, we need to 'tell' **MARK** to derive average values for each year. *Note*: make sure that the radio-button for '**`Revised model-averaging variance formulas`**' is checked.



How? Well, the first step is to think back to what the PIM for a time-dependent model looks like. Recall from earlier in this chapter that the PIMs for time-dependent recapture, but no difference between colonies, looked like:

15	16	17	18	19	20	21
	16	17	18	19	20	21
		17	18	19	20	21
			18	19	20	21
				19	20	21
					20	21
						21

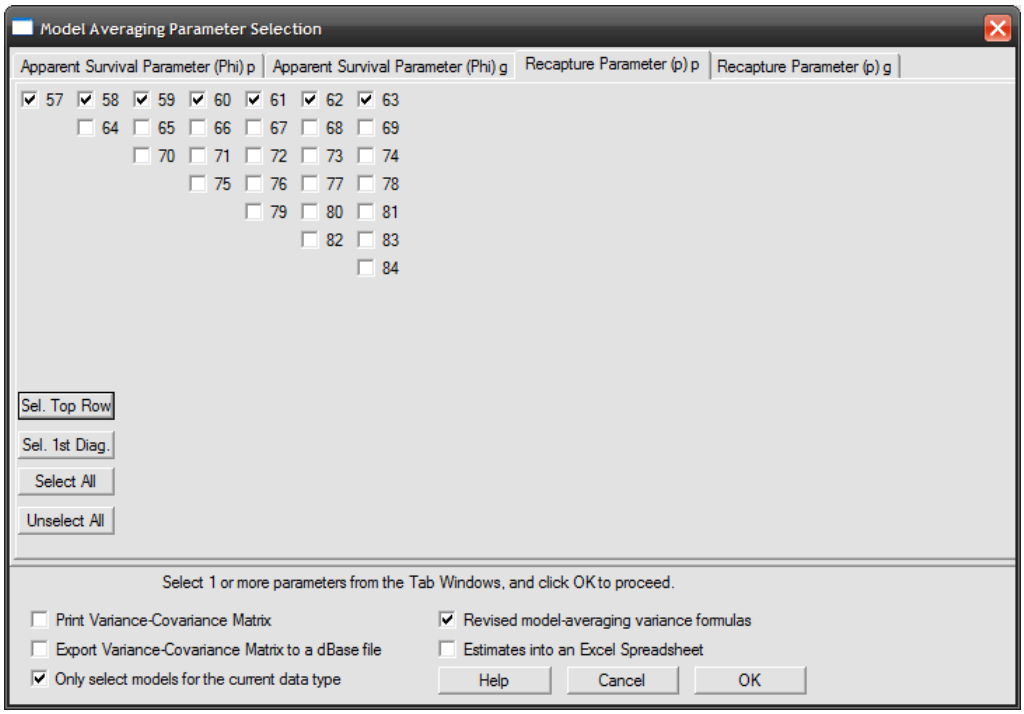
recapture 'poor'

15	16	17	18	19	20	21
	16	17	18	19	20	21
		17	18	19	20	21
			18	19	20	21
				19	20	21
					20	21
						21

recapture 'good'

Within each column (corresponding to a particular year) the index values in the PIM are the same. In other words, the survival probability in a given year does **not** depend on the year in which an individual was first marked and released (i.e., does not depend on its cohort). Thus, within the triangular matrix in the model averaging window, it doesn't matter which element of a given column you 'check', since all elements within a column are equivalent for our models!

Thus, checking each of the elements in the first row



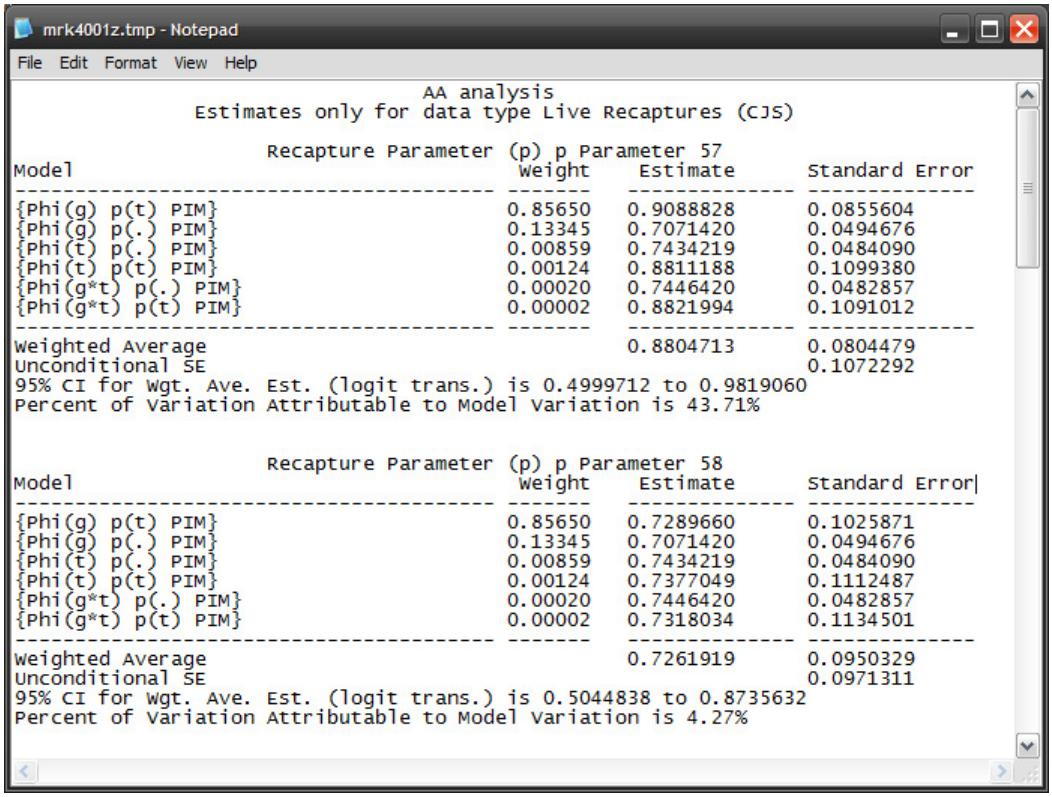
will yield exactly the same results as clicking any one element in each of the columns. Having said that, it is probably a good idea to be somewhat systematic about things (in this case, perhaps by checking each of the elements in the first row of the matrix) - it will help you keep track of things later on.

Once you've specified the cells for the recapture parameter for the poor colony, do the same for the good colony, by clicking on the tab for that parameter, and again checking one element of each column. Once you've done so, you're ready to click the '**OK**' button to start the averaging. Before you do, note that in the lower left hand corner, an option to 'only select models for the current data type' is checked by default. This option will make more sense once we've introduced the idea of switching data types, but we're a fair way from that point at this stage.

So, go ahead and click the '**OK**' button. You'll see **MARK** very quickly jump back to the results browser, and scroll through each of the models in the model set. Once it has done so, it will spawn the notepad, and present you with the 'averaged' results. The first 2 years from the output are shown at the top of the next page.

Note that the output is arranged by the index value you selected from the model averaging window. The first year corresponds to index value 57. Thus, you will need to 'keep track' of which index value corresponds to which year. For each parameter, each of the models in the candidate model set is listed, along with the model weight, the estimate from that model, and the standard error of the estimate. At the bottom of the estimate and SE columns are the 'averaged' values. For parameter 57 (which corresponds to the recapture probability for the poor colony on the first occasion), the model averaged value is 0.88047. The weighted SE is 0.08045. This is followed by something called the '*unconditional* SE', which is somewhat larger (numerically) than the weighted SE. Below the unconditional SE is a 95% CI for the model weighted average, and a statement concerning the percentage of the variation attributable to model variation (43.71% for the present example). We'll deal with the distinction between the two

SE's, and the variation due to model variation, in a moment. Note that the model averaged value of 0.88047 is somewhat lower than the estimate from the single most parsimonious model (0.9089). That is because it has been 'weighted down' by the other models in the candidate model set. Note also that only models with an AIC weight > 0 are shown (since only models with an AIC weight > 0 contribute to the weighted average).



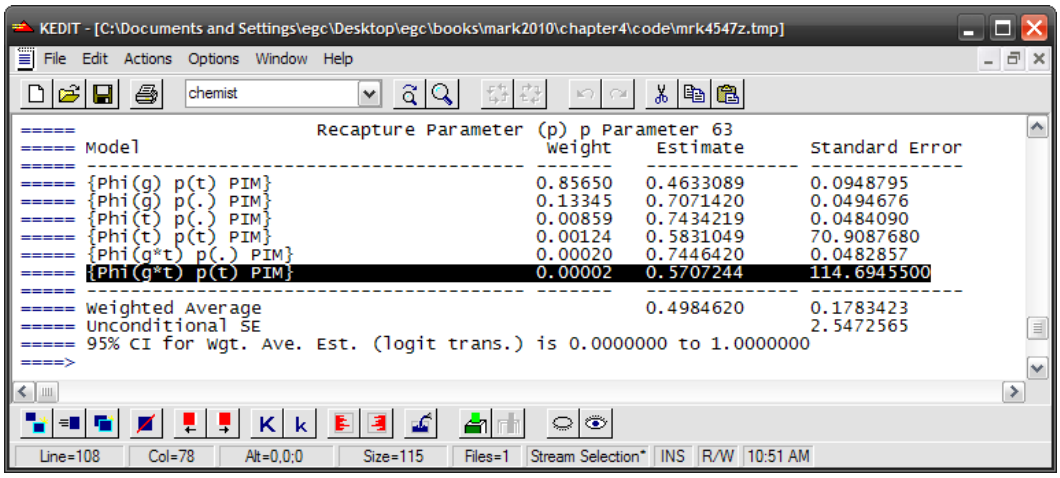
Thus, the 'best estimate' (i.e., the one we might use, or report) for recapture probability for the poor colony on the first occasion is 0.88047.

If we tabulate our model averaged estimates for all of the recapture parameters, we get:

year	*poor colony*	*good colony*
2	0.880471	0.880471
3	0.726192	0.726192
4	0.561341	0.561341
5	0.69944	0.69944
6	0.837205	0.837205
7	0.838891	0.838891
8	0.498462	0.498462

In this case, there are no differences between the poor and good colonies (for a simple reason: none of our models that had any appreciable support in the data had a colony effect for recapture!). Regardless, it is important to look at the model averaged estimates for all groups, even in situations (such as this one) where the most parsimonious models indicate no colony (group) differences.

There is one additional point we need to make here - if we look at the calculated SE's for the model-averaged estimates (the mechanics of how these are calculated is introduced in the next section), we see that for the most part they look quite 'reasonable', with one notable exception: for the final encounter probability (p_8), the estimated SE is 2.547 (with an associated 95% CI of 0 → 1). Clearly, there is a problem here. If you look closely at the model averaging output for p_8 (below), you'll see that the 'culprit' is the extremely high conditional standard errors for models $\{\varphi_t p_t\}$ and $\{\varphi_{g*t} p_t\}$.



With a bit of thought – and perhaps a peek at the reconstituted estimates for both models – you might be able to guess the underlying reason. The problem is that both of these models are fully time-dependent for both parameters, and as such, the terminal φ and p parameters are not separately identifiable. One of the characteristics of such 'confounded' parameters is extremely large (i.e., implausible) SE's. In such cases there is no absolute rule on how to proceed, but we suggest the following strategy: (i) if models with confounded parameters - or models with parameters which are poorly estimated given the data - having little → no support in the data, then it is generally acceptable to drop those models from the candidate model set, and re-run the averaging. However, (ii) if the 'problem models' have appreciable support in the data, you'll need to be more careful. You might choose simply to average only those 'well-estimated' parameters (for instance, in our example, $p_2 \rightarrow p_7$, leaving out p_8), but you need to first confirm that those models aren't well-supported simply because of the poorly estimated parameters. Again, there is no perfect solution.

What about survival? The model averaged estimates for survival are shown below:

year	*poor colony*	*good colony*
1	0.577354	0.768136
2	0.575062	0.765835
3	0.575294	0.766077
4	0.575638	0.766456
5	0.575610	0.766379
6	0.576261	0.766969
7	0.572892	0.763680

Recall that our 2 most parsimonious models (comprising ~ 99% of the support in the data) had a colony effect, but no time-dependence. We see from the model average values that there is little annual

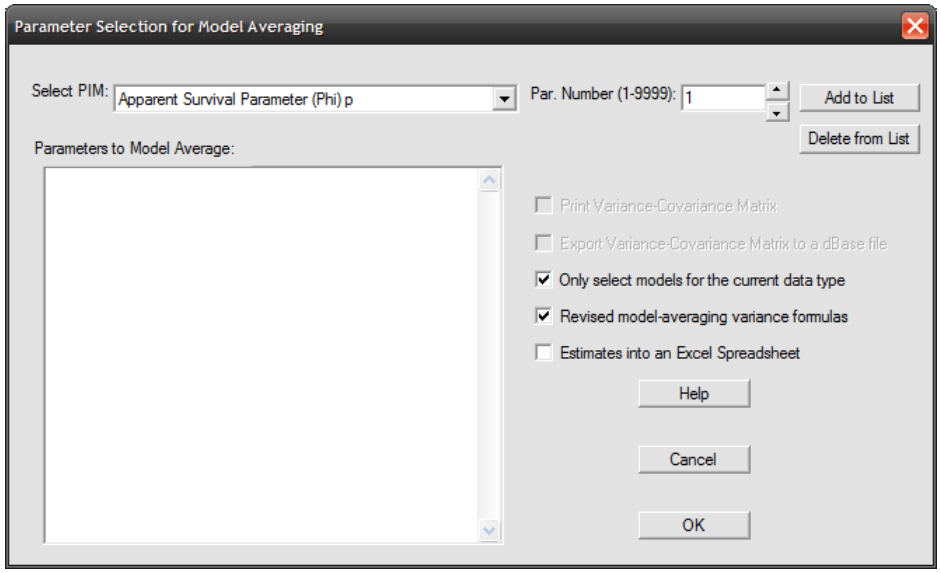
variation in the estimates - not surprising given that the models with any time-dependence had very little support in the data.

We will revisit model averaging again – it's a very important concept, since it relates directly to the important issue of 'model selection uncertainty'.

begin sidebar

Non-interactive model averaging

Some problems are too large for the interactive interface (which we just introduced) for specifying the parameters to be model averaged. An option is available in the '**File | Set Preferences**' window to change the default interface to a less interactive mode. To see how it works, select the '**non-interactive**' option in the preferences window, and restart **MARK**. Pull up the same analysis we used to demonstrate the interactive model averaging window (the *Apus apus* analysis). Again, select '**Output | Model averaging | Real**'. This will bring up the '**Non-interactive model averaging window**'.



As with the interactive model-averaging described earlier, make sure that the radio-button for '**Revised model-averaging variance formulas**' is checked.

Along the top of this window, you'll see that there is a pull-down menu, which lets you select which of the parameters you want to average (for this example, there are 4 parameters you could select among: φ_g, φ_p, p_g and p_p). To the right of this pull-down menu is a box where you select the index number of the parameter you want to average. Note that it defaults to 1-9999. However, as soon as you click inside this box, it will update to indicate the range of index values used in the data set you're analyzing (in this example, 1-28).

Now, suppose you want to model average parameters $1 \rightarrow 7$, which correspond to $\varphi_1 \rightarrow \varphi_7$ for the good colony. You simply toggle through the numbers $1 \rightarrow 7$, selecting the '**Add to list**' button for each number in turn. As with the interactive model averaging window, you can output various aspects of the averaging (e.g., into an Excel spreadsheet), simply by selecting the appropriate radio button on the right-hand side. Note that two of the options (for printing and exporting the variance-covariance matrix) are greyed out until you select at least two parameters for averaging.

end sidebar

4.5.1. Model averaging: deriving SE and CI values

In the preceding section, we noted that two different SE's for a model averaged parameter estimate were given by **MARK**: a weighted average SE (in effect, the average of the individual model SE's weighted by their respective AIC weights), and the 'unconditional SE'. These in turn were followed by a 95% CI, and a statement concerning the proportion of the variation due to model selection uncertainty. Why two different SE's? Which one is the 'right one' to report? Where does the 95% CI come from? And what does 'model selection uncertainty' refer to in the context of model averaging?

In general, the precision of an estimator should ideally have 2 *variance components*: (1) the *conditional* sampling variance, given $\widehat{\text{var}}(\hat{\theta}_i \mid M_i)$ given model i, and (2) variation associated with *model selection uncertainty*. Buckland *et al*. (1997) provide an effective method to estimate an estimate of precision that is **not** conditional on a particular model (the estimator was subsequently revised in Burnham & Anderson 2004). Assume that some scalar parameter θ is in common to all models being considered in the candidate model set. [If our focus is on a structural parameter that appears only in a subset of our full set of models, then we must restrict ourselves to that subset in order to make the sort of inferences considered here about the parameter of interest.]

So, estimates of the SE for a given model are *conditional* on that model. How do we get an *unconditional* estimate of the SE for the parameter averaged over models?

From Burnham & Anderson (2004) we will take the estimated *unconditional* variance of $\hat{\theta}$ as

$$\widehat{\text{var}}(\bar{\hat{\theta}}) = \sum_{i=1}^{R} w_i \left[\widehat{\text{var}}(\hat{\theta}_i \mid M_i) + (\hat{\theta}_i - \bar{\hat{\theta}})^2 \right]$$

where

$$\bar{\hat{\theta}} = \sum_{i=1}^{R} w_i \hat{\theta}_i$$

and the w_i are the Akaike weights (Δ_i) scaled to sum to 1. The subscript i refers to the ith model. The value $\bar{\theta}$ is a weighted average of the estimated parameter θ over R models ($i = 1, 2, \ldots, R$).

This estimator of the *unconditional* variance is clearly the sum of 2 components: (1) the conditional sampling variance ($\widehat{\text{var}}(\hat{\theta}_i \mid M_i)$) and (2) a term for the variation in the estimates across the R models $(\hat{\theta} - \bar{\hat{\theta}})^2$. The estimated *unconditional* SE is given as

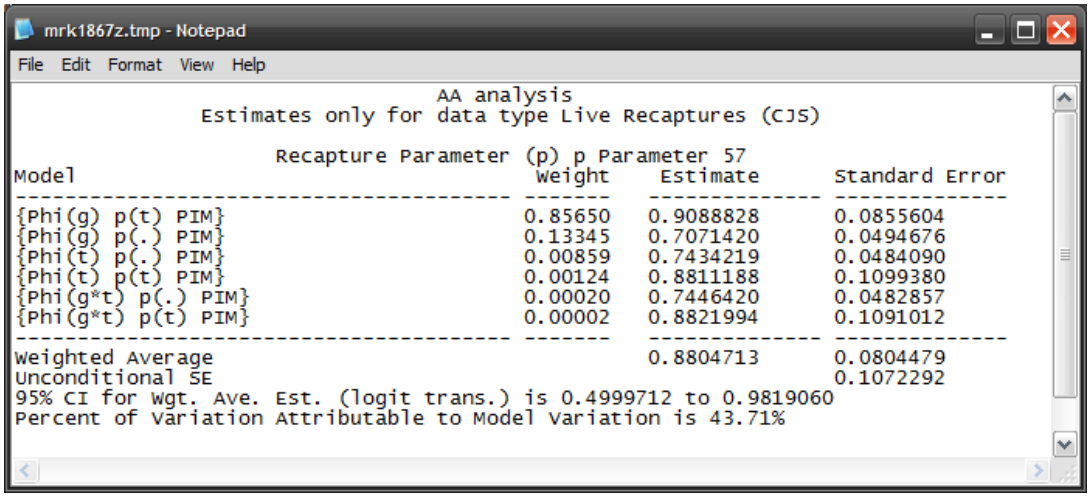
$$\widehat{\text{SE}}(\hat{\theta}) = \sqrt{\widehat{\text{var}}(\hat{\theta})}$$

It is this unconditional variance (and associated CI) that you would report, since it accounts for both conditional model-specific variation, as well as variation resulting from model selection uncertainty (i.e., among models in the candidate model set). **MARK** gives you an estimate of the proportion of the variance in the model averaged parameter estimate owing to model selection uncertainty.

Let's work through an example, using some model averaged estimates for p_2 for the poor colony (shown at the top of the next page). Start by taking a weighted average of the conditional (model-specific) SE's, weighting by the model-specific Akaike weights w_i:

$$[(0.08556 \times 0.8565) + (0.04947 \times 0.1335) + \ldots]/1.0 = 0.08045$$

which is exactly what is reported by **MARK**. Again, this is a simple weighted average, and should not be reported as the SE for the model averaged parameter estimate.



Now, let's calculate the *unconditional* SE, which **MARK** reports as 0.10723. We start by estimating the unconditional variance of the parameter as

$$\widehat{\text{var}}(\hat{\bar{\theta}}) = \sum_{i=1}^{R} w_i \left[\widehat{\text{var}}(\hat{\theta}_i \mid M_i) + (\hat{\theta}_i - \hat{\bar{\theta}})^2 \right], \quad \text{where} \quad \hat{\bar{\theta}} = \sum_{i=1}^{R} w_i \hat{\theta}_i$$

To perform this calculation, you'll need the model-specific estimates of the variance (on the normal probability scale) for the parameter. These are not given in the model averaging output (although **MARK** uses them internally during its calculations). In this example, we're considering the encounter parameter at the second occasion, for the 'poor' colony.

model	*estimate*	*variance*	*AIC weight*
$\varphi_c p_t$	0.90888	0.007321	0.85650
$\varphi_c p_.$	0.70714	0.002447	0.13345
$\varphi_t p_.$	0.74342	0.002343	0.00859
$\varphi_t p_t$	0.88112	0.012090	0.00124
$\varphi_{c*t} p_.$	0.74464	0.002332	0.00020
$\varphi_{c*t} p_t$	0.88220	0.011903	0.00002

The model averaged value for the parameter is $\hat{\bar{\varphi}} = 0.88047$. So,

$$\begin{aligned}
\widehat{\text{var}}(\hat{\bar{p}}_{2,p}) &= \sum_{i=1}^{R} w_i \left[\widehat{\text{var}}(\hat{\theta}_i \mid M_i) + (\hat{\theta}_i - \hat{\bar{\theta}})^2 \right] \\
&= 0.85650 \left[0.007321 + (0.90888 - 0.88047)^2 \right] \\
&\quad + 0.13345 \left[0.002447 + (0.70714 - 0.88047)^2 \right] \\
&\quad + \cdots \\
&\quad + 0.00002 \left[.011903 + (0.88220 - 0.88047)^2 \right] \\
&= 0.011498
\end{aligned}$$

So, our estimate of the *unconditional* variance of the encounter probability p_2 for the poor colony is 0.011498. The standard error is estimated simply as the square-root of the variance: $\sqrt{0.011498} = 0.10723$, which is what is reported by **MARK** (to within rounding error). The 95% CI reported by **MARK** is [0.5211051, 0.9803403]. Confidence intervals can also be constructed using a profile likelihood approach, but this is beyond the scope of our discussion at this point. In addition, we will leave the consideration of the reported proportion of the variation due to model selection uncertainty (and variance components analysis) to a later chapter.

begin sidebar

SE and 95% CI

The standard approach to calculating 95% confidence limits for some parameter θ is $\theta \pm (1.96 \times \mathrm{SE})$. Is this how **MARK** calculates the 95% CI on the real probability scale? Well, take the example we just considered - the estimated SE for $\hat{\varphi} = 0.88047$ was $\sqrt{0.011498} = 0.10723$. So, you might assume that the 95% CI on the real probability scale would be $0.88047 \pm (1.96 \times 0.10723) = [0.67030, 1.09064]$.

However, not only is this not what is reported by **MARK**, but it isn't even 'reasonable', since the upper CL is > 1, which is clearly not possible for a $[0, 1]$ bounded parameter. Why the difference between what **MARK** reports and what we have calculated by hand using $\theta \pm (1.96 \times \mathrm{SE})$? The difference is because **MARK** first calculates the 95% CI on the *logit* scale, before back-transforming to the real probability scale. Doing so ensures that the back-transformed CI will be $[0, 1]$ bounded. The logit scale, and back-transforming from the logit scale to the normal probability scale, is discussed in detail in Chapter 6.

end sidebar

4.6. Significance?

OK, fine. What about 'significance'? We'd hazard a guess that at this point, some (many?) of you may be wondering - 'OK - we can select a model, and can talk about the relative support of this model versus another model, we can come up with good average values, but - based on structural differences in the models, can we say anything about the significance of one or more factors?'. Often, this is the question of primary importance to the analyst - is there a 'significant' sex difference? Do the colonies differ 'significantly'? Is there evidence for a 'significant' trend over time?

Clearly, any discussion of importance, or 'significance' (in a statistical or biological context) starts with calculating the magnitude of the 'effect' - the difference in some parameter between 2 groups, or time intervals, or some other axes over which you want to characterize differences in the parameter. The question we face is 'is the difference as estimated a 'significant' difference'? Note that for the moment we've repeatedly referred to 'significance' parenthetically, since it might mean 'biological significance', or 'statistical significance', or both. And, it is critical to think critically about which context is appropriate. For example, if we simply look at the estimates for our most parsimonious model from our analysis of the swift data, we see that the estimated survival for the 'poor' colony is 0.577, and for the good colony is 0.770 - a difference of 20%. If the survival probabilities for both colonies were in fact constant over time, then we can estimate lifespan as $(1/-\ln(S))$. Thus, estimated lifespan in the good colony is 3.83 years, while in the poor colony, estimated lifespan is 1.82 years, less than 50% of the estimate for the good colony! Now, of course, whether or not $x.xx$ years (or whatever time unit is appropriate for the organism in question) is biologically significant is entirely dependent on the biology of the organism. Since this example is dealing with birds, you can rest assured that a 50% difference in expected lifespan is likely to be highly significant in the biological sense. But, this is where the biologist must use his/her judgment as to what is (or is not) a meaningful difference.

Since the effect size is 'estimated', it will have an associated uncertainty which we can specify in terms of a confidence interval (CI) (the theory and mechanics of the estimation of the effect size, and the SE

for the effect, are covered in Chapter 6). The question then becomes - what are the plausible bounds on the true effect size, and are biologically important effect sizes contained within these bounds? Suppose we consider a relative difference in survival of 15% or greater to be biologically important. Suppose the estimated effect size for the difference in survival between the colonies was 19.3%, with a CI of 1.7%-36.9% As such, we might consider the results as *statistically* 'significant', since the CI doesn't include 0, but *biologically* inconclusive, because the CI includes values *below* 15%.

It is just these sorts of questions of 'biological subjectivity' which undoubtedly contributed to the popularity of focussing on 'statistical significance', since it gives the appearance of being 'objective'. Putting aside the philosophical debate as to which type of 'significance' is more important, we'll introduce the basic mechanics for classical significance testing (in the statistical sense) as implemented in **MARK**.

4.6.1. Classical significance testing in MARK

The classical 'statistical' approach focusses on assessing the 'significance' of one or more factors on variation in a particular parameter of interest. You may recall from Chapter 1 that we can use the properties of ML estimates as the basis for a number of different 'statistical tests' (Wald, Fisher's score...) to compare the relative fit of different competing models to the data. One such test (the *likelihood ratio test*; LRT) is available in **MARK**. To apply an LRT, you take some likelihood function $\mathcal{L}(\theta_1, \theta_2, \ldots, \theta_n)$, and derive the maximum likelihood values for the various parameters θ_i. Call this likelihood $\mathcal{L}_f$. Then, fix some of the parameters to specific values, and maximize the likelihood with respect to the remaining parameters. Call this 'restricted' likelihood $\mathcal{L}_r$. The likelihood ratio test says that the distribution of twice the negative log of the likelihood ratio, i.e., $-2\ln(\mathcal{L}_r/\mathcal{L}_f)$ is approximately χ^2 distributed with with r degrees of freedom (where r is the number of restricted parameters). The LRT compares a restricted model which is '*nested*' within the full model (i.e, as is generally true with 'classical hypothesis testing', the LRT compares a pair of models).

begin sidebar

a more technical derivation of the LRT

Consider some likelihood maximized for some true parameter value θ. If we write a Taylor expansion around this value as

$$\mathcal{L}(\theta) = \mathcal{L}(\hat{\theta}) + \left(\frac{\partial \mathcal{L}}{\partial \theta}\right)_{\theta=\hat{\theta}} (\theta - \hat{\theta}) + \frac{1}{2}\left(\frac{\partial^2 \mathcal{L}}{\partial \theta^2}\right)_{\theta=\hat{\theta}} (\theta - \hat{\theta})^2 + O(\theta - \hat{\theta})^2$$

By definition,

$$\left(\frac{\partial \mathcal{L}}{\partial \theta}\right)_{\theta=\hat{\theta}} = 0$$

So, we can express the Taylor expansion as the difference between the true parameter and the estimated parameter:

$$\mathcal{L}(\theta) - \mathcal{L}(\hat{\theta}) = \frac{1}{2}\left(\frac{\partial^2 \mathcal{L}}{\partial \theta^2}\right)_{\theta=\hat{\theta}} (\theta - \hat{\theta})^2 + O(\theta - \hat{\theta})^2$$

$$-2[\mathcal{L}(\theta) - \mathcal{L}(\hat{\theta})] = \left(-\frac{\partial^2 \mathcal{L}}{\partial \theta^2}\right)_{\theta=\hat{\theta}} (\theta - \hat{\theta})^2 + O(\theta - \hat{\theta})^2$$

Dropping off the residual term,

$$-2[\mathcal{L}(\theta) - \mathcal{L}(\hat{\theta})] \cong \left(-\frac{\partial^2 \mathcal{L}}{\partial \theta^2}\right)_{\theta=\hat{\theta}} (\theta - \hat{\theta})^2$$

Now, consider a simple binomial process, where we aim to estimate the parameter $\hat{p}$. Then, we can write

$$-2[\mathcal{L}(p) - \mathcal{L}(\hat{p})] \cong \left(-\frac{\partial^2 \mathcal{L}}{\partial p^2}\right)_{p=\hat{p}} (p - \hat{p})^2$$

Now, recall from Chapter 1 (and basic common sense) that the MLE for $\hat{p}$ is (n/N) (say, n successes in N trials). Also recall that $-\partial^2 \mathcal{L}/\partial p^2$ is an estimate of the variance of $\hat{p}$. Then,

$$\begin{aligned} -2[\mathcal{L}(p) - \mathcal{L}(\hat{p})] &\cong \left(-\frac{\partial^2 \mathcal{L}}{\partial p^2}\right)_{p=\hat{p}} (p - \hat{p})^2 \\ &= \frac{1}{\widehat{\text{var}}(\hat{p})} \cdot \left(p - \frac{n}{N}\right)^2 = \frac{p - E(\hat{p})}{\widehat{\text{var}}(\hat{p})^2} \end{aligned}$$

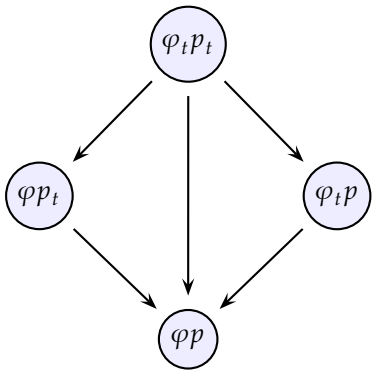
Now, some classical results show that as $N \to \infty$, then

$$\hat{p} \to N\left(E(\hat{p}), \sqrt{\text{var}(\hat{p})}\right) \quad \text{and} \quad \frac{\left(p - E(\hat{p})\right)^2}{\widehat{\text{var}}(\hat{p})} \to N(0,1)^2 = \chi^2_1$$

In other words, the parameter estimate is asymptotically normal convergent as sample size increases, and (more to the point here), that $-2(\mathcal{L}(p) - \mathcal{L}(\hat{p}))$ (i.e., the deviance) is χ^2 distributed. This is a very convenient result - it says that as the sample size n approaches ∞, the test statistic $-2\ln(\Lambda)$ will be asymptotically χ^2 distributed with degrees of freedom equal to the difference in dimensionality (number of parameters) of the two models being compared. This means that for a great variety of hypothesis, a practitioner can take the likelihood ratio Λ, algebraically manipulate Λ into $-2\ln(\Lambda)$, compare the value of $-2\ln(\Lambda)$ given a particular result to the χ^2 value corresponding to a desired statistical significance, and create a reasonable decision based on that comparison.

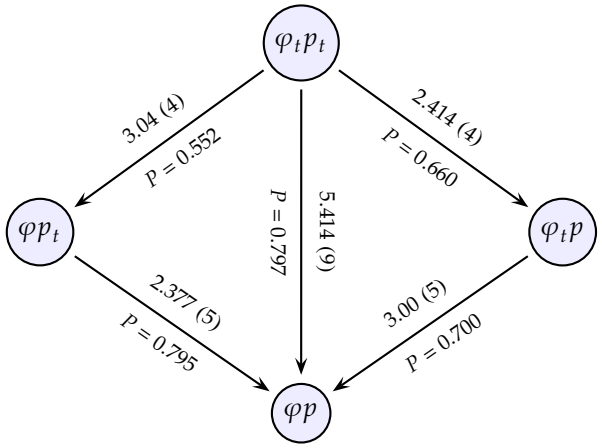
end sidebar

In practical terms, the first step in using the LRT with models fit using **MARK** is to determine which models are nested. While this is not always as straightforward as it seems (see the - `sidebar` - a few pages ahead), it is relatively straightforward for our present example. Consider the following figure, which represents the hierarchical relationship of the 4 models we fit to the male Dipper data.



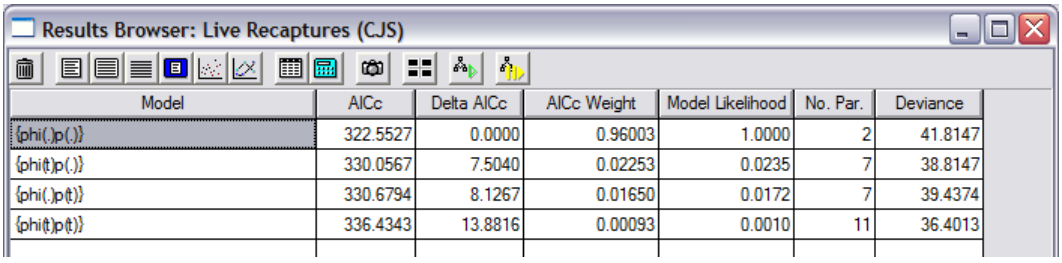
In this figure, 'nested' models are connected by the arrows. The direction of the arrows leads from a given model to the model 'nested' within it. Any two 'nested' models can be compared statistically using a likelihood ratio test. Provided that the reduced (less parameterized) model is satisfactory, the difference in deviances between two nested models is distributed as a χ^2 statistic with n degrees of freedom, where n is the difference in the number of parameters between the two models.

We re-draw this figure at the top of the next page, showing the differences in deviance among 'nested' models, and the difference in the number of parameters, we obtained from our Dipper analysis.



The 'significance' of these differences (in the traditional sense) can be estimated from any standard χ^2 table, or directly using **MARK**. Recall from Chapter 3 that in **MARK** you can request a likelihood ratio test (LRT) between any two models, using the '**Tests**' menu. However, **MARK** doesn't 'know this', and performs LRT for **all** models with unequal numbers of parameters, and outputs results from all these comparisons. Clearly, the 'unequal number of parameters' criterion is not a particularly good one, so you'll need to pay attention. A significant difference between models means two things: (1) that there is a significant increase in deviance with the reduction in the number of parameters, such that the reduced model fits significantly less well, and (2) the parameter(s) involved contribute significantly to variation in the data. As we can see from the figure, there is no significant difference in model fit (difference in deviance) between the most parameterized model (the CJS model $\varphi_t p_t$) and any of the 3 other models. Thus, any of the 3 other models would be a 'better model' than the CJS model, since they fit the data equally well (statistically), but require fewer parameters to do so (i.e., are more parsimonious).

From the preceding figure and from the AIC$_c$ values tabulated in the results browser


Results Browser: Live Recaptures (CJS)

Model	AICc	Delta AICc	AICc Weight	Model Likelihood	No. Par.	Deviance
{phi(.)p(.)}	322.5527	0.0000	0.96003	1.0000	2	41.8147
{phi(t)p(.)}	330.0567	7.5040	0.02253	0.0235	7	38.8147
{phi(.)p(t)}	330.6794	8.1267	0.01650	0.0172	7	39.4374
{phi(t)p(t)}	336.4343	13.8816	0.00093	0.0010	11	36.4013

we see that the most parsimonious model overall is model $\{\varphi_{.} p_{.}\}$ - i.e., the model where both survival and recapture probability are constant over years.

However, before we go any further, what hypotheses have we just tested? Consider the test of the CJS model $\{\varphi_t p_t\}$ versus $\{\varphi_\cdot p_t\}$. In this comparison we are testing the following 'hypothesis': that there is significant variation in survival over time. We are comparing the fit of a model where survival is allowed to vary over time $\{\varphi_t p_t\}$ to one where it doesn't $\{\varphi_\cdot p_t\}$. Since the fit of these two models is not 'significantly' different in the classical statistical sense ($\chi^2 = 3.05$, $df = 4$, $P = 0.552$), we might state that 'there is no evidence at a nominal $\alpha = 0.05$ level of significant annual variation in survival'.

begin sidebar

Which models are nested?

While the preceding example was simple enough, determining which models are nested is not always trivial. Here, we take a slightly more extended look at 'nestedness' and linear models.

Let's begin with addressing the question of why aren't models $\{\varphi_\cdot p_t\}$ and $\{\varphi_t p_\cdot\}$ in the preceding example nested? The easiest way to resolve which models of the models in this example are nested, and which aren't, is to try to answer the following question: 'would starting model **A** be equivalent to reduced model **B** if you eliminated one or more of the factors from model **A**?' If so, then model **B** is 'nested' within model **A**. For example, if we start with model $\{\varphi_t p_t\}$ (model **A**), we want to know if model $\{\varphi_t p_\cdot\}$ (model **B**) is nested within it. So, what happens if you 'remove one or more of the factors from model **A**'? Well, in this case we see that if we eliminate 'time' from capture in model **A**, then model **A** is transformed into model **B**. Thus, we can say that model **B** $\{\varphi_t p_\cdot\}$ is nested within model **A** $\{\varphi_t p_t\}$.

However, now compare models $\{\varphi_\cdot p_t\}$ and $\{\varphi_t p_\cdot\}$. If we consider these models as **A** and **B** respectively, we see that there is no simple transformation of model **A** into model **B**; we would have to drop the time-dependence from the recapture model, and add time to the survival model, to make models **A** and **B** equivalent. Since nesting requires only addition or subtraction of parameters (but not both), then these models are clearly not nested.

But, these examples are very simple. What about situations where 'nestedness' is not so obvious. For example, are the models $\{Y = x\}$ and $\{Y = \ln(x)\}$ nested? Clearly, we need a more general set of rules. Let's start by considering models which *are* nested.

> **nested models**: *Two models are nested if one model can be reduced to the other model by imposing a set of linear restrictions on the vector of parameters.*

For example, consider models f and g, which we'll assume have the same functional form and error structure. For convenience, we'll express the data as deviations from their means (doing so eliminates the intercept from the linear model, since it would be estimated to be 0). These two models differ then only in terms of their regressors. In the following

$$f: \; Y = \beta_1 x_1 + \epsilon_0$$
$$g: \; Y = \beta_1 x_1 + \beta_2 x_2 + \epsilon_1$$

the model f is nested within model g because by imposing the linear restriction that $\beta_2 = 0$, model g becomes model f.

What about non-nested models? Things get a bit more complex here, but we'll operationally define non-nested models as

> **non-nested models**: *Two models are non-nested, either partially or strictly* (discussed below), *if one model cannot be reduced to the other model by imposing a set of linear restrictions on the vector of parameters*

Examples of non-nested models include (but not limited to):

- **No linear restriction possible to reduce on model to another**

  Consider

$$f: Y = \beta_1 x_1 + \beta_2 x_2 + \epsilon_0$$
$$g: Y = \beta_2 x_2 + \beta_3 x_3 + \epsilon_1$$

  Models f and g are non-nested because even if we impose the restriction on model g that $\beta_3 = 0$, model g does not become model f.

  In fact, in this example, models f and g are *partially non-nested*, because they have one variable in common (x_2). If the two models didn't share x_2, then they would be *strictly non-nested*.

  However, you need to be somewhat careful in defining models as strictly non-nested. There are, in fact, two cases where models with different sets of regressors may not be strictly non-nested.

  Consider the following two models:

$$f: Y = \beta_1 x_1 + \epsilon_0$$
$$g: Y = \beta_2 x_2 + \epsilon_1$$

  If either β_1 or β_2 equals zero, then the models are nested. This is trivially true. Less obvious, perhaps, is the situation where one of more of the explanatory variables in one model may be written as a linear combination of the explanatory variables in the second model. For example, given the two models

$$f: Y = \beta_1 x_1 + \epsilon_0$$
$$g: Y = \beta_2 x_2 + \epsilon_1$$

  consider a third model h where

$$h: Y = \beta_3 x_3 + \epsilon_2 = \beta_1 x_1 + \beta_2 x_2 + \epsilon_2$$

  Then, perform the following hypothesis tests: model h versus model f (i.e., $\beta_2 = 0$ versus $\beta_2 \neq 0$), and model h versus model g (i.e., $\beta_1 = 0$ versus $\beta_1 \neq 0$).

- **Different functional forms used in two models**

  The following are clearly different function forms

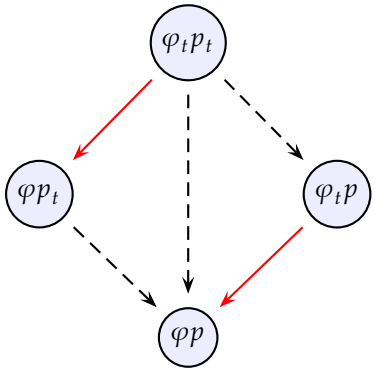
$$f: Y = X\beta + \epsilon$$
$$g: \ln(Y) = \ln(X)\gamma + \mu$$

end sidebar

4.6.2. Some problems with the classical approach...

Seems straightforward, right? Ah, but there are at least three 'mechanical' problems with this approach (we'll ignore the more philosophical questions concerning the apparent subjectivity of specifying the nominal α-level for evaluating the 'significance' of an LRT, although it is clearly an important part of the larger debate). First, the LRT is only strictly appropriate when the models being compared are nested. For non-nested models, you can use either the AIC, or approaches based on some sort of resampling ('bootstrapping') approach (it is worth noting that the AIC was introduced in the seminal Lebreton *et al*. 1992 paper in part as a 'mechanical' solution to just this problem). The question of what constitutes 'nested' was addressed in the preceding `- sidebar -`.

Second, if you look (again) at the hierarchial diagram shown below



you should notice that there are in fact 2 different pairs of models (red arrows) we could compare (using an LRT) to test for annual variation in survival: $\{\varphi_t p_t\}$ versus $\{\varphi_\cdot p_t\}$, or model $\{\varphi_t p_\cdot\}$ versus $\{\varphi_\cdot p_\cdot\}$. In both cases, we are testing for significant annual variation in survival (i.e., φ_t). Do the two different tests give the same results? Do both tests lead to the same conclusion about annual variation in apparent survival, φ?

We'll explore this question using live encounter data contained in the file `LRT-demo.inp` – 5 sampling occasions. Here are the results of fitting models $\{\varphi_t, p_t\}$, $\{\varphi_t, p_\cdot\}$, $\{\varphi_\cdot p_t\}$ and $\{\varphi_\cdot p_\cdot\}$ to these data:

Results Browser: Live Recaptures (CJS)

Model	AICc	Delta AICc	AICc Weight	Model Likelihood	No. Par.	Deviance	-2Log(L)
{Phi(.) p(t) PIM}	849.3549	0.0000	0.52042	1.0000	5	13.3161	839.2382
{Phi(t) p(.) PIM}	850.5384	1.1835	0.28798	0.5534	5	14.4996	840.4217
{Phi(t) p(t) PIM}	852.2986	2.9437	0.11944	0.2295	7	12.1577	838.0798
{Phi(.) p(.) PIM}	853.3062	3.9513	0.07217	0.1387	2	23.3609	849.2830

Our interest lies in comparison of model $\{\varphi_t, p_t\}$ with model $\{\varphi_\cdot p_t\}$, versus a comparison of model $\{\varphi_t, p_\cdot\}$ with model $\{\varphi_\cdot p_\cdot\}$. Using '**`Tests | LR Tests`**', we generate the following test results:

```
Reduced Model            General Model              Chi-sq.  df  Prob.
------------------------ ------------------------ ---------- --- ------
{Phi(.) p(t) PIM}        {Phi(t) p(.) PIM}           -1.184   0 ******
{Phi(.) p(t) PIM}        {Phi(t) p(t) PIM}            1.158   2 0.5604
{Phi(.) p(.) PIM}        {Phi(.) p(t) PIM}           10.045   3 0.0182
{Phi(t) p(.) PIM}        {Phi(t) p(t) PIM}            2.342   2 0.3101
{Phi(.) p(.) PIM}        {Phi(t) p(.) PIM}            8.861   3 0.0312
{Phi(.) p(.) PIM}        {Phi(t) p(t) PIM}           11.203   5 0.0475
```

The model comparisons of interest are shaded in the preceding figure. For the comparison of the general model $\{\varphi_t, p_t\}$ with nested (reduced) model $\{\varphi_\cdot p_t\}$, the LRT yields a $\chi^2_2 = 1.158$, which is not close to significant at the nominal $\alpha = 0.05$ level ($P = 0.5604$). versus a comparison of model $\{\varphi_t, p_\cdot\}$ with model $\{\varphi_\cdot p_\cdot\}$. Taken alone, this would suggest no strong evidence for annual variation in apparent survival.

However, now consider the comparison of the general model $\{\varphi_t, p_.\}$ with nested (reduced) model $\{\varphi_. p_.\}$. Here, the LRT yields a $\chi^2_3 = 8.861$, which is in fact significant at the nominal $\alpha = 0.05$ level ($P = 0.0312$). So, this comparison would suggest that there *is* evidence for annual variation in apparent survival, opposite to what we concluded in the first analysis!

The observation that the different model comparisons yielded very different results, and thus different conclusions, is clearly a problem. In fact, for any typical candidate model set, there are likely to be > 1 pairs of nested models which could be compared using a LRT to test the same hypothesis. And there is a fair likelihood that in many cases, these LR tests will yield different, often contradictory results (as in our present example).

Thus, the obvious question is: which of the possible nested comparisons for a given hypothesis is the 'right one' to use? A commonly suggested approach (which is not without its critics) is to start from the most parsimonious acceptable model still containing the effect you want to test, and then use the LRT to test the nested model without this factor. You can use AIC (or sequential LRT tests where appropriate) to identify the model which has the fewest parameters while still fitting the data and containing the factor you are interested in. The advantage of using this model is that tests are generally most powerful in a 'parsimonious context'. In our example, model $\{\varphi_t, p_.\}$ is more parsimonious than model $\{\varphi_t, p_t\}$, and thus we might conclude that the LRT between model $\{\varphi_t, p_.\}$ and nested (reduced) model $\{\varphi_. p_.\}$ is the more powerful of the two comparisons, and thus, supporting a conclusion of 'significant' annual variation in apparent survival.

However, there is still at least one final problem that does not seem to have a satisfactory solution. Specifically, the presence of 'nuisance' parameters (for example, the recapture parameter p in a mark-recapture analysis) is not accounted for in LRTs. For example, consider the following sets of models: $\{\varphi_t, p_t\}$ vs $\{\varphi_. p_t\}$, $\{\varphi_t, p_.\}$ vs $\{\varphi_. p_.\}$, and $\{\varphi_t p_t\}$ vs $\{\varphi_. p_.\}$. In all cases, the structure of the φ parameter for the second model of each pair is nested within the structure for φ in the first model. As such, considering only φ, it appears that all three pairs of models represent nested models which could be contrasted using the classical LRT.

However, notice that the structure for p for the pairs of models differs. Consider the first two pairs - in the first pair, the recapture parameter is modeled as time-dependent (p_t), whereas in the second pair, it is modeled as time-invariant ($p_.$). And, consider the third pair, where the parameterization for p differs between the two models (the second model being nested within the first for both p and φ). Which of these tests is most appropriate to test the basic hypothesis φ_t vs $\varphi_.$? Here, the differences in the models being compared concern the 'other parameters' in the model, and the approach recommended for identifying the model structure for the most robust test for φ (discussed above) clearly doesn't apply.

4.6.3. 'Significance' of a factor using AIC

Despite several difficulties with the classical approach to testing for 'statistical significance', there would seem to be one singular advantage relative to multimodel inference based on an information theoretic index like the AIC or BIC. Namely, that there is a relatively straightforward way (caveats notwithstanding) to give some sort of statement about the 'significance' (importance) of some factor(s). Is there something equivalent that can be done with the information theoretic approach?

Burnham & Anderson have noted that assessment of the relative importance of variables has often been based only on the best model (e.g., often selected using a stepwise testing procedure of some sort). Variables in that best model are considered 'important', while excluded variables are considered 'not important'. They suggest that this is too simplistic. Importance of a variable can be refined by making inference from all the models in the candidate set. Akaike weights are summed for all models containing predictor variable (i.e., factor) x_j, $j = 1, \ldots, R$. Denote these sums as $w_{+(j)}$. The predictor variable with

the largest predictor weight, $w_{+(j)}$, is estimated to be the most important, while the variable with the smallest sum is estimated to be the least important predictor.

Can we be somewhat more 'formal' about this? Consider the following example - a simple linear model with three regressors, x_1, x_2 and x_3. The objective was to examine the eight possible models consisting of various combinations of these regressors.

In the following table are the possible models, along with hypothetical AIC weights (in the table, a '1' indicates that x_i is in the model; otherwise, it is excluded).

x_1	x_2	x_3	w_i
0	0	0	0.00
1	0	0	0.10
0	1	0	0.01
0	0	1	0.05
1	1	0	0.04
1	0	1	0.50
0	1	1	0.15
1	1	1	0.15

The selected best model has a weight of only 0.5 (suggesting strong model selection uncertainty). However, the sum of the weights for variable x_1 across all models containing x_1 is 0.79. This is evidence of the importance of this variable, across all eight of the model considered. Variable x_2 was not included in the selected best model, but this does not mean that it is of no importance (which might be the conclusion if you made inference only on the K-L best model). Actually, its relative weight of evidence support is 0.35. Finally, the sum of AIC weights for x_3 is 0.85.

Thus, the evidence for the importance of variable x_3 is substantially more than just the weight of evidence for the best model. We can order the three predictor variables in this example be their estimated importance: x_3, x_1, x_2 with importance weights of 0.85, 0.79, and 0.35, respectively. This basic idea extends to subsets of variables. For example, we can judge the importance of a pair of variables, *as a pair*, by the sum of the AIC weights of all the models that include that pair of variables. Similar procedures apply when assessing the relative importance of interaction terms.

To see this approach in application to a mark-recapture analysis, let's reconsider the results from the swift analysis:

Results Browser: Live Recaptures (CJS)

Model	AICc	Delta AICc	AICc Weight	Model Likelihood	No. Par.	Deviance	-2Log(L)
{phi(c)p(t)}	369.8080	0.0000	0.85650	1.0000	9	111.6644	350.8705
{phi(c)p(.)}	373.5263	3.7183	0.13345	0.1558	3	128.1990	367.4051
{phi(t)p(.)}	379.0123	9.2043	0.00859	0.0100	8	123.0601	362.2662
{phi(t)p(t)}	382.8807	13.0727	0.00124	0.0014	13	115.7384	354.9446
{phi(c*t)p(.)}	386.4962	16.6882	0.00020	0.0002	15	114.7094	353.9155
{phi(c*t)p(t)}	391.4103	21.6023	0.00002	0.0000	20	107.5633	346.7694

We notice that the two most parsimonious models in the candidate model set have colony differences in the survival parameters - model $\{\varphi_c p_t\}$ and model $\{\varphi_c p_.\}$ have virtually all of the support in the

data. Model $\{\varphi_c p_t\}$ has approximately 6-7 times more support than model $\{\varphi_c p_.\}$. Moreover, the top two models, comprising (0.857+0.133) = 99% of the support in data, both have φ_c for in common for the survival parameter. Meaning, only models with a colony effect on the apparent survival rate have any appreciable support in the data. At this point, we might conclude that there is considerable evidence of a difference in survival between the 2 colonies. What about using cumulative support over all models in the model set? The summed AIC weights for `colony`, `time`, and `colony*time` for survival are: 0.9896, 0.0098, and 0.0007, respectively. Clearly, there is overwhelming support for a `colony` effect.

As suggested by Anderson & Burnham, summing support over models is regarded as superior to making inferences concerning the relative importance of variables based only on the best model. This is particularly important when the second or third best model is nearly as well supported as the best model or when all models have nearly equal support. (There are 'design' considerations about the set of models to consider when a goal is assessing variable importance. For example, it is important to achieve a balance in the number of models that contain each variable. In the swift analysis, each main effect (colony, time) was present in an equal number of models).

While this approach seems to have some merit, there is by no means consensus that this is the 'best approach', or that it 'works' in some cases – see for example Murray & Conner (2009).* One aspect of design of the candidate model set which seems to be particularly important is that the model set by 'symmetrical' or 'balanced' with respect to each factor (i.e., that the model set has roughly the same number of models with, and without a particular factor). This is not always easy, or obvious how to accomplish (how do you balance models for interaction terms?). See Doherty, White & Burnham (2010)† for a discussion of this and related issues. Clearly, the assessment of 'relative importance' of variables in complex models is a 'work in progress'.

4.7. LRT or AIC?

Right about now, you're probably thinking a couple of things. First, we've covered a lot of ground. Indeed, so we have. We've seen how to build and fit various models to our data using **MARK**. Second, we've given some thought to model selection – the use of LRT and AIC (or BIC), counting parameters, and 'hypothesis testing'. We've also consider the important idea of 'model averaging'. You're also probably thinking (hopefully) thats it's about time for us to make broad, categorical suggestions about what to do - AIC/BIC or LRT? Significance test, or effect size?

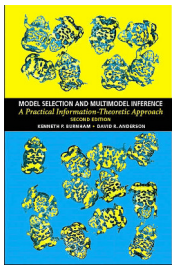
Alas, prepare to be disappointed. While we have our personal opinions, **MARK** itself is 'politically neutral' on this one - you can choose to adopt an 'information theoretic' approach, or invoke a classic LRT approach (but **not** combinations of the two – you have to pick either of the 'two roads', and not mix and match the two), and talk about the significance of an effect based on a nominal *P*-value.

The whole issue of whether or not to use *P*-values, and 'classical hypothesis testing' is the subject of much debate in the literature. We **strongly** suggest you read some of the recent work on the AIC (and related issues) by David Anderson and Ken Burnham.

* Murray & Conner (2009) Methods to quantify variable importance: implications for the analysis of noisy ecological data. *Ecology* **90**:348-355.

† Doherty, White & Burnham (2010) Comparison of model building and selection strategies. *Journal of Ornithology*, DOI 10.1007/s10336-010-0598-5

In particular, you really should work through their detailed but highly approachable text*:



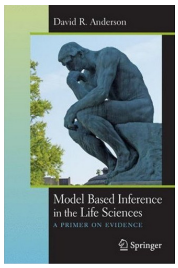
Model Selection and Multi-Model Inference (2nd Edition) - Ken Burnham and David Anderson. 2002. Spring-Verlag. 496 pages.

and (for the more faint of heart) the various 'gentler' (i.e., more accessible) introductions they have published, including the following two papers:

Anderson, D. R. & Burnham, K. P. (1999) Understanding information criterion for selection among capture-recapture or ring recovery models. *Bird Study*, **46**, 14-21.

Anderson, D. R. & Burnham, K. P. (1999) General strategies for the analysis of ringing data. *Bird Study*, **46**, 261-270.

and, quite recently, a very useful (and very accessible) primer by Anderson:



Model Based Inference in the Life Sciences: A Primer on Evidence - David Anderson. 2007. Springer. 184 pages.

4.8. Summary

This brings us to the end of Chapter 4. And you thought all you had to do was select some pre-defined models, and **MARK** would do the rest for you, eh? In this chapter, we looked at the basic mechanics of using **MARK** to test several models. We've looked at the problem of staggered entry of marked individuals into the population (and how this leads logically to the triangular parameters structures - the PIMs). We've also considered (at an introductory level) the mechanics and theory of statistical tools: the LRT and the AIC. In the next chapter, we'll consider the issue of goodness of fit testing – an essential step in model selection.

* Well, perhaps Chapter 7 is a 'bit dense'. However, this book is where you should start if you want a more complete understanding of the derivation of the AIC, which we've only summarized earlier in this chapter.

Addendum: counting parameters

Although **MARK** does a good job of counting parameters, it is important that you understand how the model structure determines the number of parameters that are theoretically estimable. What **MARK** does is indicate (report) how many parameters are estimable, given the model, and the data. **MARK** does not indicate how many parameters are theoretically estimable, given the structure of the model. On occasion, there are discrepancies between the two.

There are 2 reasons why a particular model parameter might not be estimable. The first is because the parameter may be confounded with 1 or more other parameters in the model. An example is the last φ and p parameters in a time-specific Cormack-Jolly-Seber model, where only the product of φ and p can be estimated, but not the unique values of each. In this case, the parameters are not identifiable because of the *structure* of the model. This is referred to as *intrinsic non-identifiability*. The second situation arises either because the data are inadequate, or as an artifact of the parameter being 'poorly estimated' near either the 0 or 1 boundaries. This is referred to as *extrinsic non-identifiability*. If a parameter is extrinsically non-identifiable because of 'problems' with the data, then you may need to manually increase the number of estimated parameters **MARK** reports (given the data) to the number that *should* have been estimated (if there had been sufficient data). While there has been significant progress in formal 'analytical' analysis of intrinsic identifiability, these methods are complex, and do not apply generally to problems related to inadequate data or parameters estimated near the boundary. A numerical approach based on data cloning which can be used generally to help identify parameters that are not estimable is available in **MARK** is presented in Appendix F.

intrinsically non-identifiable parameters - an ad hoc approach

While there are methods (formal, numerical) for identifying intrinsically non-identifiable parameters, it is important that you develop an 'intuitive understanding' of the how such parameters arise in the first place. Let's assume that our data are 'good' - there are no 'structural problems' and that the only remaining task is to determine which parameters are separately identifiable. We'll concentrate on the 4 models we've examined in this chapter. We'll introduce an approach which is generally useful, if a bit cumbersome. In future chapters, where we explore significantly more complex models, we'll comment as needed on how the number of parameters was determined. Our most complex model in this chapter is the CJS model - complete time-dependence in both survival and recapture. In many ways, the most fundamental difficulty in counting parameters in general is nicely contained in this model, so it is a good starting point.

However, before we dive in, consider a much simpler situation. Consider the case of only 2 occasions, a release occasion, where newly marked individuals are released, and a single recapture occasion. This situation is common in short-term studies. In general, under this sampling scheme, what is done is to express the proportion of the individuals marked and released on the first occasion captured on the second occasion as a measure of the 'survival probability'. This fraction, also known as the 'return probability', is still widely found in the literature.

Unfortunately, naive use of return probability poses real problems, since, in fact, it does not necessarily estimate survival probability at all. As noted in Lebreton *et al*. (1992), the number of individuals seen on the second occasion is the result of 2 events, not one; the frequency of individuals seen again on the second occasion is defined by the product of the number released on occasion 1 (R_1) times the probability of surviving to occasion 2 (φ_1), times the probability of being seen at occasion 2 given that it is in fact alive at occasion 2 (the recapture probability, p_2). Since the value of φ_1 and p_2 can vary between 0 and 1, the observed number of individuals at occasion 2 could reflect an infinite set of different

combinations of either survival or recapture probability. For example, suppose 100 individuals are marked and released at occasion 1, and 50 of these marked individuals are seen subsequently at occasion 2. The return probability is (50/100) or 0.5. However, does this really mean that 'survival' is 50%? Not necessarily. What it means is that $(100 \times \varphi_1 p_2) = 50$, or $(\varphi_1 p_2) = 0.5$. As you quickly see, there is an infinite set of combinations of φ_1 and p_2 which, when multiplied together, lead to the product 0.5. Thus, we can't necessarily say that 'survival' is 0.5, merely that the combined probability of surviving and being recaptured is 0.5. In other words, with only 2 occasions, the survival and recapture probabilities are not 'individually identifiable' - we cannot derive estimates for both parameters separately.

What do we need to do? Well, in order to separately derive estimates for these parameters, we need more information. We need at least one additional recapture occasion. The reason is fairly obvious if you look at the capture histories. As per Lebreton *et al*. (1992), let '1' represent when an individual is captured at a particular occasion, and '0' represent when it is not captured. With only 2 occasions and individuals released marked only on the first occasion, only 2 capture histories are possible: `10` and `11`. As we just observed, with only two captures we can estimate only the product of survival and recapture. What about three occasions? As noted in Chapter 1, under this sampling scheme, at least 4 capture histories are possible for individuals marked on the first occasion:

encounter history	*probability*
`111`	$\varphi_1 p_2 \varphi_2 p_3$
`110`	$\varphi_1 p_2 (1 - \varphi_2 p_3)$
`101`	$\varphi_1 (1 - p_2) \varphi_2 p_3$
`100`	$1 - \varphi_1 p_2 - \varphi_1 (1 - p_2) \varphi_2 p_3$

The capture histories are given with the probability statements which, when multiplied by the number released at occasion 1, define the number of individuals with a given capture history expected at occasion 3. Concentrate for the moment on the third capture history in the table: '`101`'. You can see that there is a fundamental difference in this capture history from the one preceding it (where individuals are seen on each occasion). For capture history '`101`', individuals were released on occasion 1, not seen on occasion 2, but were seen again on occasion 3. What does this sort of individual tell us? Well, clearly, if the individual was seen on occasion 3, then it must have been alive on occasion 2. The fact that we didn't see the individual at occasion 2 allows us to estimate the recapture probability, since recapture probability is merely the probability of seeing an animal at a particular occasion given that it is alive. Thus, because we have information from the third occasion, we can separately estimate the survival and recapture probabilities φ_1 and p_2 respectively. Specifically,

$$\frac{N_{111}}{N_{101}} = \frac{\varphi_1 p_2 \varphi_2 p_3}{\varphi_1 (1 - p_2) \varphi_2 p_3} = \frac{p_2}{1 - p_2}$$

Of course, **MARK** shields you from the complexities of the actual estimation itself, but in a very broad sense, it is the presence of '`101`' individuals along with the other capture histories that allows us to estimate survival and recapture rate separately.

But, it is important to note that we can't separately estimate **all** the parameters. Consider for instance φ_2 and p_3. Can we separate them? No! In fact, the product of these two parameters is completely analogous to a return probability between occasions 2 and 3. If we wanted to separate these 2 parameters, we'd need a fourth occasion, and so on. Thus, in such a model where both survival and recapture probability are time-dependent, the terminal parameters are not individually identifiable - all we can do is estimate the product of the 2. Lebreton *et al*. (1992) refer to this product term as β_3.

Thus, we can re-write our table, and the probability statements, as:

encounter history	*probability*
111	$\varphi_1 p_2 \beta_3$
110	$\varphi_1 p_2 (1 - \beta_3)$
101	$\varphi_1 (1 - p_2) \beta_3$
100	$1 - \varphi_1 p_2 - \varphi_1 (1 - p_2) \beta_3$

Now, we come to the original question: how many parameters do we have? In this case, with 3 occasions, and time-dependence in both survival and recapture, we have 3 estimable parameters: φ_1, p_2, and β_3. Do we always have a 'beta' parameter - a terminal product that cannot be separated into its component survival and recapture elements? The answer is, 'no'. Whether or not you have a 'beta' term depends upon the structure of your model. We can demonstrate this by going back to the 4 models used in this chapter. We start with the fully time-dependent CJS model. Clearly, from the preceding discussion, you might expect that there is likely to be a 'beta' term, since we have time-dependence for both parameters. Your intuition is correct. How can we count them? While there are a number of possible schemes you could use to count parameters (including rote memory of certain algebraic relationships between the number of time units and the number of parameters for a given type of model - see Table 7 in Lebreton *et al*. 1992), we prefer a more cumbersome, but fairly fool-proof way of counting them without resorting to memorization.

To use this approach, simply do the following. For a given model, write out all the saturated capture histories, and their associated probability statements, for each cohort. A 'saturated capture history' is the capture history where the individual was seen on each occasion following its release. In our Dipper example, there are 7 occasions, so our table of saturated capture histories, and substituting $\beta_7 = \varphi_6 p_7$, the associated probability statements, would look like :

encounter history	*probability*
1111111	$\varphi_1 p_2 \varphi_2 p_3 \varphi_3 p_4 \varphi_4 p_5 \varphi_5 p_6 \beta_7$
0111111	$\varphi_2 p_3 \varphi_3 p_4 \varphi_4 p_5 \varphi_5 p_6 \beta_7$
0011111	$\varphi_3 p_4 \varphi_4 p_5 \varphi_5 p_6 \beta_7$
0001111	$\varphi_4 p_5 \varphi_5 p_6 \beta_7$
0000111	$\varphi_5 p_6 \beta_7$
0000011	β_7

Now, all you need to do is count how many unique parameters there are. A parameter is unique if it occurs at least once in any of the probability statements. If you count the unique parameters in this table, you will see that there are 11 of them: 5 survival probabilities (φ_1 to φ_5), 5 recapture probabilities (p_2 to p_6), and one 'beta' term, β_7, the product of $\varphi_6 p_7$. Note, that this is only a technique to help you count the number of 'potentially' identifiable parameters - this does **not** necessarily mean that all of them are estimable. That is determined by the data. We introduce an approach (based on 'data cloning') for handling this issue in Appendix F.

Now, a fair question at this point is 'why do we need to write out the saturated capture histories and the probability statements for all cohorts, since we could have used just the first cohort to count unique parameters?'. Well, the answer is, in this case, you really didn't need to. However, as you will see, this approach is useful and necessary for more complicated models. We introduce it now just to get you in the habit.

Let's consider the next two models: $\{\varphi_t p_\cdot\}$ and $\{\varphi_\cdot p_t\}$. From the results browser, we see that both models have 7 parameters. Let's confirm this. Again, we use the 'saturated capture histories approach'. Start with the model $\{\varphi_t p_\cdot\}$:

encounter history	*probability*
1111111	$\varphi_1 p \varphi_2 p \varphi_3 p \varphi_4 p \varphi_5 p \varphi_6 p$
0111111	$\varphi_2 p \varphi_3 p \varphi_4 p \varphi_5 p \varphi_6 p$
0011111	$\varphi_3 p \varphi_4 p \varphi_5 p \varphi_6 p$
0001111	$\varphi_4 p \varphi_5 p \varphi_6 p$
0000111	$\varphi_5 p \varphi_6 p$
0000011	$\varphi_6 p$

Now, in this case, we do not have a terminal β term. The terminal product is $\varphi_6 p_\cdot$. Are both parts separately estimable? Yes. Since the constant recapture probability occurs at each occasion, we can use the information from preceding occasions to estimate the value of p. And, if we know the recapture probability p, then we can estimate any of the survival probabilities, including φ_6. Specifically, φ_6 is estimated as

$$\hat{\varphi}_6 = \frac{\hat{\beta}_7}{\hat{p}}$$

under the assumption that $p_7 = p$ (this is clearly an untestable assumption). Thus, we have 7 identifiable parameters: 6 survival rates (φ_1 to φ_6) and 1 recapture probability (p). For the model $\{\varphi_\cdot p_t\}$, we have the same situation (7 estimable parameters) but in reverse. Finally, for our model $\{\varphi_\cdot p_\cdot\}$ (constant survival and recapture), there are only two estimable parameters. By now, you should be able to prove this to yourself. Try it!

How does MARK count parameters?

Needless to say, **MARK** uses a more 'technical' approach to counting the number of estimable parameters than the *ad hoc* approach described above. In Chapter 1, we considered the derivation of the MLE and the variance for a simple example involving a model with only 2 parameters: φ and p. The likelihood for the particular example was given as

$$\ln \mathcal{L}(\varphi, p) = 7 \ln(\varphi p \varphi p) + 13 \ln(\varphi p (1 - \varphi p)) + 6 \ln(\varphi (1 - p) \varphi p) + 29 \ln(1 - \varphi p - \varphi (1 - p) \varphi p)$$

We first derived the Hessian $\mathbf{H}$ as the matrix of second partial derivatives of the likelihood $\mathcal{L}$ with respect to the parameters φ and p):

$$\mathbf{H} = \begin{bmatrix} \frac{\partial^2 \mathcal{L}}{\partial \varphi^2} & \frac{\partial^2 \mathcal{L}}{\partial \varphi \partial p} \\ \frac{\partial^2 \mathcal{L}}{\partial p \partial \varphi} & \frac{\partial^2 \mathcal{L}}{\partial p^2} \end{bmatrix}$$

Next, we evaluated the Hessian at the MLE for φ and p (i.e., we substituted the MLE values for our parameters - $\hat{\varphi} = 0.6648$ and $\hat{p} = 0.5415$ - into the Hessian), which yielded the information matrix $\mathbf{I}$

$$\mathbf{I} = \begin{bmatrix} -203.06775 & -136.83886 \\ -136.83886 & -147.43934 \end{bmatrix}$$

The negative inverse of the information matrix ($-\mathbf{I}^{-1}$) is the variance-covariance matrix of the parameters φ and p

$$-\mathbf{I}^{-1} = -\begin{bmatrix} -203.06775 & -136.83886 \\ -136.83886 & -147.43934 \end{bmatrix}^{-1}$$
$$= \begin{bmatrix} -0.0122 & 0.0181 \end{bmatrix}$$

While deriving the variance-covariance matrix is obviously the basis for estimating parameter precision, there is further utility in the information matrix: skipping the theory, the *effective rank* of the information matrix is an estimate of the *maximum* number of estimable parameters (but this does not account for confounded parameters). The effective rank of

$$-\mathbf{I}^{-1} = \begin{bmatrix} 0.0131 & -0.0122 \\ -0.0122 & 0.0181 \end{bmatrix}$$

is 2, meaning, we have 2 estimable parameters (which by now we know to be true for this model).

What is the *effective rank* of a matrix? Technically, the rank of a matrix (or a linear map, to be complete) is the dimension of the range of the matrix (or the linear map), corresponding to the number of linearly independent rows or columns of the matrix (or to the number of nonzero singular values of the map). The details can be found in any introductory text on linear algebra, but the basic idea is easily demonstrated.

Consider the following (4×4) matrix

$$\begin{bmatrix} 2 & 4 & 1 & 3 \\ -1 & -2 & 1 & 0 \\ 0 & 0 & 2 & 2 \\ 3 & 6 & 2 & 5 \end{bmatrix}$$

We see that the second column is twice the first column, and that the fourth column equals the sum of the first and the third. Thus, the first and the third columns are *linearly independent*, so the rank of **A** is two.

What about a less obvious case? For example, suppose we re-write the likelihood in terms of time-specific φ and p parameters:

$$\ln \mathcal{L}(\varphi_1, \varphi_2, p_2, p_3) = 7 \ln(\varphi_1 p_2 \varphi_2 p_3) + 13 \ln(\varphi_1 p_2 (1 - \varphi_2 p_3)) + 6 \ln(\varphi_1 (1 - p_2) \varphi_2 p_3) + 29 \ln(1 - \varphi_1 p_2 - \varphi_1 (1 - p_2) \varphi_2 p_3)$$

We use **MARK** to find the MLE estimates as: $\hat{\varphi}_1 = 0.6753$, $\hat{p}_2 = 0.5385$, and $\hat{\varphi}_2 = \hat{p}_3 = 0.5916$. Now, what is important here is that the terminal β_3 term is estimated as the product of φ_2 and p_3 - in fact, the estimates of φ_2 and p_3 could be **any** values from $0 \rightarrow 1$, as long as the product $(\varphi_2 p_3) = 0.5916^2 = 0.3500$, (where $0.3500 = \hat{\beta}_3 = (\hat{\varphi}_2 \hat{p}_2)$. This becomes important later on.

For now, though, let's concentrate on parameter counting. First, we derive the Hessian, which for this model $\{\varphi_t p_t\}$ is given as

$$\mathbf{H} = \begin{bmatrix} \frac{\partial^2 \mathcal{L}}{\partial \varphi_1^2} & \frac{\partial^2 \mathcal{L}}{\partial \varphi_1 \partial \varphi_2} & \frac{\partial^2 \mathcal{L}}{\partial \varphi_1 \partial p_2} & \frac{\partial^2 \mathcal{L}}{\partial \varphi_1 \partial p_3} \\ \frac{\partial^2 \mathcal{L}}{\partial \varphi_2 \partial \varphi_1} & \frac{\partial^2 \mathcal{L}}{\partial \varphi_2^2} & \frac{\partial^2 \mathcal{L}}{\partial \varphi_2 \partial p_2} & \frac{\partial^2 \mathcal{L}}{\partial \varphi_2 \partial p_3} \\ \frac{\partial^2 \mathcal{L}}{\partial p_2 \partial \varphi_1} & \frac{\partial^2 \mathcal{L}}{\partial p_2 \partial \varphi_2} & \frac{\partial^2 \mathcal{L}}{\partial p_2^2} & \frac{\partial^2 \mathcal{L}}{\partial p_2 \partial p_3} \\ \frac{\partial^2 \mathcal{L}}{\partial p_3 \partial \varphi_1} & \frac{\partial^2 \mathcal{L}}{\partial p_3 \partial \varphi_2} & \frac{\partial^2 \mathcal{L}}{\partial p_2 \partial p_3} & \frac{\partial^2 \mathcal{L}}{\partial p_3^2} \end{bmatrix}$$

Again, the *variances* of the parameters are along the diagonal, and the *covariances* are off the diagonal.

Next, we substitute in the MLE estimate for our 4 parameters. While we have unique estimates for $\hat{\varphi}_1$ and $\hat{p}_2$, what about the terminal $\hat{\beta}_3$ term? If we use the values **MARK** reports ($\hat{\varphi}_2 = \hat{p}_3 = \hat{\beta}_3 = 0.5916$), then the resulting information matrix is singular (meaning: we can't invert it to derive the variance-covariance matrix). Is this a problem? Well, yes and no. A problem clearly if we want to estimate the variance-covariance matrix for our parameters (which we obviously want to do for any model). But, if the information matrix is singular, what can you do? Well, what if instead of $\hat{\varphi}_2 = \hat{p}_3 = 0.5916$, we instead had used $\hat{\varphi}_2 = 0.3500, \hat{p}_3 = 1.0$ (such that $\hat{\beta}_3$ still equals 0.3500). Again, remember that the estimates of φ_2 and p_3 could be **any** value from $0 \rightarrow 1$, as long as the product $(\varphi_2 p_3) = 0.5916^2 = 0.3500$ (as noted above). Substituting these values into the Hessian yields the information matrix

$$\mathbf{I} = \begin{bmatrix} -108.121 & -48.143 & -67.802 & -16.850 \\ -48.143 & -147.026 & 22.871 & -51.459 \\ -67.802 & 22.871 & -117.246 & 8.005 \\ -16.850 & 51.459 & 8.005 & -18.011 \end{bmatrix}$$

If we take the negative inverse of this matrix, we see that the variance-covariance matrix is

$$-\mathbf{I}^{-1} = \begin{bmatrix} 0.0235 & -0.0082 & -0.0156 & -0.0056 \\ -0.0082 & 318.191 & 0.0054 & -909.091 \\ -0.0156 & 0.0054 & 0.0191 & 0.0075 \\ -0.0056 & -909.091 & 0.0075 & 2597.417 \end{bmatrix}$$

Obviously, the variances for φ_2 and p_3 are 'wonky' (from the Latin). We discussed earlier how this can (on occasion) be used as a rough diagnostic to when parameters are inestimable.

But, our main objective here is to determine how *many* parameters are estimable? If we take the rank of this information matrix, we get 4 - which is correct, because in effect we've 'manually separated' the elements of the $\hat{\beta}_3$ term. What if we had calculated the rank of the matrix substituting $\hat{\varphi}_1 = 0.6753$, $\hat{p}_2 = 0.5385$, and $\hat{\varphi}_2 = \hat{p}_3 = 0.5916$? We noted already that the information matrix using these values is singular, but...what about the rank? In fact, if we take the rank of the information matrix, we get 3, which matches the number of estimable parameters.

But, how many parameters are actually estimated given the data? In **MARK**, computation of the Hessian is performed with a finite central difference approximation for the second derivatives (i.e., the second derivatives are estimated *numerically*, not *analytically*). How does this work? Well, first define

$\mathcal{L}_{0,0}$ as the log likelihood computed for the maximum likelihood estimates of a β_i and β_j. Further define $\mathcal{L}_{1,0}$ as the log likelihood computed with β_i incremented by the amount h, and $\mathcal{L}_{2,0}$ as the log likelihood computed with β_i incremented by the amount $2h$. Similarly, $\mathcal{L}_{-2,0}$ is the log likelihood computed with decremented by the amount $2h$. Using this notation, the second partial of the log likelihood for β_i is computed as:

$$\frac{\partial^2 \mathcal{L}_{0,0}}{\partial \beta_i^2} = \frac{1}{12h^2}\left(-\mathcal{L}_{2,0} + 16\mathcal{L}_{1,0} - 30\mathcal{L}_{0,0} + 16\mathcal{L}_{-1,0} - \mathcal{L}_{-2,0}\right)$$

and the joint partial of the log likelihood for β_i and β_j is computed as:

$$\frac{\partial^2 \mathcal{L}_{0,0}}{\partial \beta_i \partial \beta_j} = \frac{1}{4h^2}\left(-\mathcal{L}_{1,1} + \mathcal{L}_{1,-1} - \mathcal{L}_{-1,1} + \mathcal{L}_{-1,-1}\right)$$

Given the number of function evaluations needed to compute these derivatives, it is obvious why the computation of the variance-covariance matrix takes so long to complete once the optimizations have completed*. However, a precise calculation of the information matrix is needed, not only to provide an estimate of the variance-covariance matrix of the β estimates, but also to compute the estimated number of parameters.

To invert and also compute the rank of the Hessian, a numerical approach based on a singular-value decomposition is computed (for you techno-philes - using the DSVDC algorithm of Dongarra *et al*. 1979). This algorithm returns an array of singular values sorted into descending order of the same length as the number of rows or columns of the Hessian. These values are then 'conditioned' by dividing through by the maximum value, so that now the values range from a maximum of 1 to a value equivalent to zero *if there are more β parameters in the model than can be estimated* (this is important).

The trick is to determine whether the smallest value(s) of the conditioned singular values is (are) actually zero. Two rules are applied to make this decision. First, a *threshold value* is computed (and printed in the full output file) that is a guess at what the minimum conditioned value would be smaller than if there were more betas than can be estimated. This threshold is based on the number of parameters used in the optimization and the value of h used to compute the Hessian. The precision of the numerical estimates in the Hessian is a function of h, as well as the number of columns in the design matrix (the number of β values).

Using this threshold value, all values of the conditioned singular values vector that are smaller than the threshold are considered to be parameters that have not been estimated. Conversely, all values of the conditioned singular value vector that are greater than the threshold are considered to be parameters that were estimated, and are part of the parameter count.

The threshold condition may suggest that all of the β values were parameters that were estimated, i.e., the smallest conditioned singular value is greater than the threshold. An additional test is performed to evaluate whether some of the β parameters were not actually estimated. The ratio of consecutive values in the sorted conditioned singular value array is used to identify large jumps in the singular values. Typically, the ratios of consecutive values decline slowly until a large gap is reached where parameters are not estimated.

As an example, consider the following portion of the output for the global model (i.e., model $\{\varphi_t p_t\}$) of the male Dipper data, where the last φ and last p is identifiable only as a product (i.e., they are not separately estimable - a point we've made before). Thus, instead of the 12 parameters that you might

* All experienced **MARK** users have learned to be patient as the variance-covariance matrix is calculated, especially for complex models

have initially assumed are estimable, only 11 are actually estimated. The conditioned singular values are identified as the **S** vector, with the condition index being the smallest value of the **S** vector.

Here are the relevant sections from the full **MARK** output:

```
 Threshold =  0.2600000E-06     Condition index =  0.7594599E-08
Conditioned S Vector {phi(t)p(t)}:
  1.000000     0.9393990     0.8540481     0.7733464     0.6198060
 0.3407142     0.2980727     0.2426806     0.2241094     0.6495305E-01
 0.6369173E-01 0.7594599E-08
Number of Estimated Parameters {phi(t)p(t)} = 11
```

In this output, 11 parameters are reported to have been estimated. The last value of the sorted **S** vector is smaller than the threshold, and thus considered to have *not* been estimated. Also, this example illustrates the major jump in the ratio of consecutive **S** values from the 11th to the 12th values that corresponds to loss of estimability, i.e., a ratio of (0.7594599*E*–08/0.6369173*E*–01). In problems where all of the **S** values are greater than the threshold, the ratio of consecutive values is computed for all consecutive pairs, and a ratio > 500 is assumed to have identified the breakpoint between estimable and non-estimable parameters.

An important point is how the link function can play into this process. In the above example, the sin link was used, so that parameters on their boundaries were still considered estimable. In contrast, with the logit link, parameters on their boundary often appear to have not been estimated. The following output is for the identical model, but now run with a logit link.

```
 Threshold =  0.2600000E-06     Condition index =  0.2266159E-09
Conditioned S Vector {phi(t)p(t)}:
  1.000000     0.8414351     0.7774529     0.7095102     0.6336003
 0.1953204     0.8798561E-01 0.8716604E-01 0.8589297E-01 0.5509756E-01
 0.5461554E-07 0.2266159E-09
Number of Estimated Parameters {phi(t)p(t)} = 10
```

Because p_3 is estimated at its upper boundary of 1, a second value in the **S** vector is now less than the threshold, and so the parameter count is 10 instead of 11. In this case, the threshold should have been < 0.5461554*E*–07 but > 0.2266159*E*–09 to have achieved a correct parameter count.

The reason that the male p_3 estimate appears to be singular and not estimable is that the β estimate for this parameter was 23.19, which appears numerically (for the log likelihood) to have almost a zero first and second derivative for this parameter. In fact, a graph of the β values over the range 20 to 25 would suggest the log likelihood is flat. As a result, this parameter is considered to not have been estimated, even though it actually was estimated. The user must correct the parameter count manually. Alternatively, use the sin link to avoid problems with highly parameterized models (we'll talk a lot more about link functions in Chapter 6).

You may notice that **MARK** gives you the option of choosing between two different procedures to estimate the variance-covariance matrix of the estimates. The first is the inverse of the Hessian matrix obtained as part of the numerical optimization of the likelihood function. This approach is not reliable, and should only be used when you are not interested in the standard errors, and already know the number of parameters that were estimated. The only reason for including this method in the program is that it is the fastest – no additional computation is required for the method.

The second method (the default) computes the information matrix directly using central difference approximations to the second partial derivatives. This method (labeled the **2ndPart** method) provides the most accurate estimates of the standard errors, and is the default and preferred method.

Because the rank of the variance-covariance matrix is used to determine the number of parameters that were actually estimated, using different methods will sometimes result in a different number of parameters estimated, which can have important implications for model selection.

CHAPTER 5

Goodness of fit testing...

In the preceding chapter, we took our first look at a fundamental topic – comparing models. In particular, we considered the different 'paradigms' of AIC, or LRT. More generally, however, these approaches both rest fundamentally on issues of fit - how well does a particular model fit the data. This is a very important consideration, regardless of which 'flavor' of model selection you prefer both AIC comparisons, and LRT, require assessment of model fit.

In this chapter we will provide a brief introduction to this very important topic - goodness of fit testing (GOF). All of the models and approaches we have discussed so far make very specific assumptions (concerning model fit) that must be tested before using **MARK** - thus, as a first step, you need to confirm that your starting (general) model adequately fits the data, using GOF tests. We will make frequent reference to this, directly or indirectly, at several points throughout the book.

There are a number of ways in which GOF testing can be accomplished, and a variety of GOF procedures have been implemented in several different CMR software applications. For example, programs **RELEASE**, **SURVIV**, **JOLLY**, and **JOLLYAGE** all provide GOF statistics for various models. Some applications do not provide any 'built-in' GOF testing. As a starting point, we will assert that there are two primary purposes for GOF testing.

The first, which we've already noted, is that it is a necessary first step to insure that the most general model in your candidate model set (see Chapter 4) adequately fits the data. Comparing the relative fit of a general model with a reduced parameter model provides good inference only if the more general model adequately fits the data.

However, suppose you construct a candidate model set, based on *a priori* hypotheses concerning the data in hand. This model set contains at least one 'general' model, containing enough parameter structure so that, you believe, it will fit the data. Suppose however, it does not - suppose you have a means of assessing GOF, and that based on this 'test' you determine that the general model does not adequately fit the data. What next? Well, in addition to providing a simple 'yes/no' criterion for fit, the GOF testing procedure can in itself reveal interesting things about your data. While significant lack of fit of your general model to your data is in some senses a nuisance (since it means you need to carefully reconsider your candidate model set), in fact the lack of fit forces you to look at, and think about, your data more carefully than you might have otherwise - the key question becomes - *why* doesn't the model fit the data? The answers to this question can sometimes be extremely valuable in understanding your problem.

© Cooch & White (2014)

What do we mean by 'lack of fit'? Specifically, we mean that the arrangement of the data do not meet the expectations determined by the assumptions underlying the model. In the context of simple mark-recapture, these assumptions, sometimes known as the 'CJS assumptions' are:

1. every marked animal present in the population at time (i) has the same probability of recapture (p_i)
2. every marked animal in the population immediately after time (i) has the same probability of surviving to time (i+1)
3. marks are not lost or missed.
4. all samples are instantaneous, relative to the interval between occasion (i) and (i+1), and each release is made immediately after the sample.

We will generally assume that assumptions 3 and 4 are met (although we note that this is not always a reasonable assumption. For example, neck collars, commonly used in studies of large waterfowl, have a significant tendency to 'fall off' over time). It is assumptions 1 and 2 which are typically the most important in terms of GOF testing.

In this chapter, we will look at GOF testing in two ways. First, we shall discuss how to do basic GOF testing in program **MARK**, using a parametric bootstrapping approach. Then, we show how to use program **MARK** to call another, vintage program (**RELEASE**) to more fully explore potential reasons for lack of fit for the CJS model only. Then, we introduce two newer approaches to estimating lack of fit. We finish by discussing how to accommodate lack of fit in your analyses.

5.1. Conceptual motivation - 'c-hat' ($\hat{c}$)

GOF testing is a diagnostic procedure for testing the assumptions underlying the model(s) we are trying to fit to the data. To accommodate (adjust for, correct for...) lack of fit, we first need some measure of how much extra binomial 'noise' (variation) we have. The magnitude of this overdispersion cannot be derived directly from the various significance tests that are available for GOF testing, and as such, we need to come up with some way to quantify the amount of overdispersion. This measure is known as a variance inflation factor, or (hereafter, $\hat{c}$, or phonetically, 'c-hat').

We start by introducing the concept of a *saturated model*. The saturated model is loosely defined as the model where the number of parameters equals the number of data points or data structures. As such, the fit of the model to the data is effectively 'perfect' (or, as good as it's going to get).

begin sidebar

saturated models in MARK

In the following, the method used to compute the saturated model likelihood is described for each type of data. This is the method used when no individual covariates are included in the analysis. Individual covariates cause a different method to be used for any data type.

Live Encounters Model. For the live encounters model (Cormack-Jolly-Seber model), the encounter histories within each attribute group are treated as a multinomial. Given n animals are released on occasion i, then the number observed for encounter history j [n_j] divided by n is the parameter estimate for the history. The $-2\ln(\mathcal{L})$ for the saturated model is computed as the sum of all groups and encounter histories. For each encounter history, the quantity $n_j * \ln[n_j/n]$ is computed, and then these values are summed across all encounter histories and groups.

Dead Encounters Model - Brownie. The method used is identical to the live encounters model. For this type of data, the saturated model can be calculated by specifying a different value in every

PIM entry. The resulting $-2\ln(\mathcal{L})$ value for this model should be identical to the saturated model value.

Dead Encounters Model - Seber. For dead encounters models with S and f coding. The saturated model for this data type is the same as the usual dead encounters model.

Joint Live and Dead Encounters Model. The method used is identical to the live encounters model.

Known Fate Model. The known fate data type uses the group*time model as the saturated model. For each occasion and each group, the number of animals alive at the end of the interval divided by the number of animals alive at the start of the interval is the parameter estimate. The $-2\ln(\mathcal{L})$ value for the saturated model is the same as the $-2\ln(\mathcal{L})$ value computed for the group*time model.

Closed Captures Model. The saturated model for this type of data includes an additional term over the live encounters model, which is the term for the binomial coefficient portion of the likelihood for $\hat{N}$. For the saturated model, $\hat{N}$ is the number of animals known to be alive [M(t+1)], so the log of $\hat{N}!$ factorial is added to the likelihood for each group.

Robust Design Model. The saturated model for this data type is the same as the closed captures model, but each closed-captures trapping session contributes to the log likelihood.

Multi-strata Model. The saturated model for this data type is the same as for the live encounters model.

BTO Ring Recovery Model. The saturated model for this data type is the same as for the live encounters data.

Joint Live and Dead Encounters, Barker's Model. The method used is identical to the live encounters model.

Pradel and Link-Barker Models. These models assume that an animal can enter the study on any occasion, so the saturated model is computed with the parameter estimate as the number of animals with the encounter history divided by the total number of animals encountered. The same procedure is used for the Burnham Jolly-Seber model and the POPAN model, but because these data types include population size, complications result.

All Data Types with Individual Covariates. For any of the models with individual covariates, the sample size for each encounter history is 1. The saturated model then has a $-2\ln(\mathcal{L})$ value of zero. The deviance for any model with individual covariates is then just its $-2\ln(\mathcal{L})$ value.

end sidebar

Consider the following example of a logistic regression of some medical data. Suppose there is a sample of male and female cardiac patients. Interest is focussed on whether or not the amplitude (high or low) of a particular waveform in an electrocardiograph test (EKG) was a good predictor of cardiac disease (`0` = no disease, `1` = disease), and whether the predictions were influenced by the gender of the patient. Here are the data, presented as a frequency table:

female

	EKG	
disease	h	l
0	10	15
1	16	9

male

	EKG	
disease	h	l
0	12	11
1	9	17

If we use both `sex` and `EKG` and their interaction in the model, we will use up all the degrees of freedom. That is, we are fitting each cell in the contingency table with its own parameter, which constitutes a saturated model:

$$\texttt{disease} = \texttt{sex} + \texttt{EKG} + (\texttt{sex*EKG})$$

If we fit this model to the data (using the logistic regression program from your favorite statistics package), -2 times the model likelihood is given as $-2\ln(\mathcal{L}) = 132.604$. The AIC for this model is 140.604 (132.604+[2×4] = 140.604). Remember – the saturated model is the model where the model structure is saturated with respect to the data. In other words, it is sufficiently parameterized that every data point is effectively encompassed by the model. As such, the likelihood for the saturated model is as small as it's ever going to get.

Now, in this case, the parameter values for the terms in the saturated model are all estimable. This will not generally be the case. Moreover, in many cases, the saturated model is not a plausible, or interesting general starting model. For the moment, let's pretend that is the case here. Suppose that instead of the saturated linear model proposed above, our general starting model is

$$\texttt{disease} = \texttt{sex} + \texttt{EKG}$$

If we fit this model to the data, the model likelihood is $-2\ln(\mathcal{L}) = 136.939$, with an AIC of 142.939. As expected, the fit isn't as good as the saturated model. But, the point of interest here is – how different is the fit? The numerical difference between the likelihood for the saturated model and the general model is (136.939 − 132.604) = 4.335. The difference in the degrees of freedom (number of estimable parameters) between the two models is 1 (the interaction term).

Now, for the **key conceptual step** - the difference in fit (deviance) between the saturated model and any other model (in this case, the general model in the candidate model set) is asymptotically χ^2 distributed (at least, in theory). In **MARK**, the *deviance* (as reported in the results browser) is defined as the difference in $-2\ln(\mathcal{L})$ between the current model and the saturated model. For our example analysis, $\chi^2_1 = 4.335$ is marginally significant ($P = 0.0373$) based on a nominal $\alpha = 0.05$ level. This suggests that the general model does not quite adequately fit the data.

So, why is this important? Well, suppose we didn't know that the general model in our model set has some lack of fit to the data (relative to the saturated model), and proceeded to compare the general model with a reduced parameter model

$$\texttt{disease} = \texttt{EKG}$$

In other words, we're comparing

$$\begin{array}{rl} & \texttt{disease} = \texttt{SEX+EKG+error} \\ \textit{versus} & \underline{\texttt{disease} = \quad \texttt{EKG+error}} \\ & \texttt{SEX} \end{array}$$

which amounts to a test of the importance of `SEX` in determining the presence or absence of the cardiac disease. The likelihood for the reduced model (`disease=EKG`) is $-2\ln(\mathcal{L}) = 137.052$, with an AIC=141.052. If we use a LRT to compare the fits of the general and reduced models, we get a test statistic of $\chi^2_1 = (137.052 - 136.939) = 0.0113$, which is clearly not significant by usual statistical standards.

However, in making this comparison we've ignored the fact that our general model has marginally significant lack of fit to the data (relative to the saturated model). Does this make a difference in our analysis? In fact, the generally accepted approach to this would be to 'adjust' (correct) the likelihood of both the general model and the reduced model to account for the lack of fit between the general and saturated models.

For a correctly specified model, the χ^2 statistic (or the deviance) divided by the degrees of freedom, should be approximately equal to one. When their values are much larger than one, the assumption of

simple binomial variability may not be valid and the data are said to exhibit *overdispersion*. *Underdispersion*, which results in the ratios being less than one, occurs less often in practice.

The most common approach to correcting for overdispersion in linear models is to multiply the covariance matrix by a dispersion parameter (*note*: this approach is most robust when the sample sizes in each subpopulation in the analysis are roughly equal). In other words, we use a function of the lack of fit (typically, some function of the χ^2/df for the general model), to adjust the fit of the general and all other models in the candidate model set. For our present example, applying a χ^2/df 'correction' yields $-2\ln(\mathcal{L}) = 31.590$ for the general model, and $-2\ln(\mathcal{L}) = 31.616$ for the reduced model.

Do we need to modify the LRT in any way? In fact, the LRT, which is normally a χ^2 test between two models, is transformed into an F-test, with (df_{LRT},df_{model}) degrees of freedom:

$$F = \frac{\left(\chi^2_{LRT}/\mathrm{df}_{LRT}\right)}{\hat{c}}$$

where

$$\hat{c} \approx \frac{\chi^2}{df} = 1$$

For this example, no big difference in the subsequent LRT between the two models.

What about the AIC approach? Well, recall from Chapter 4 that the sample-size corrected AIC_c is estimated as

$$\mathrm{AIC}_c = -2\ln(\mathcal{L}(\hat{\theta})) + 2K + \left(\frac{2K(K+1)}{n-K-1}\right)$$

Do we need to adjust the AIC_c for lack of fit between the general model and the saturated model? Perhaps given the preceding discussion it is not surprising that the answer is 'yes'. We have to adjust the likelihood term, yielding the quasi-likelihood adjusted QAIC_c

$$\mathrm{QAIC}_c = \frac{-2\ln(\mathcal{L})}{\hat{c}} + 2K + \left(\frac{2K(K+1)}{n-K-1}\right)$$

where $\hat{c}$ is the measure of the lack of fit between the general and saturated models.*

Now, since

$$\hat{c} \approx \frac{\chi^2}{df} = 1$$

for a saturated model, then as the general model gets 'further away' from the saturated model, $\hat{c} > 1$. If $\hat{c} = 1$, then the expression for QAIC_c reduces back to AIC_c (since the denominator for the likelihood term disappears). If $\hat{c} > 1$, then the contribution to the QAIC_c value from the model likelihood will decline, and thus the relative penalty for a given number of parameters K will increase. Thus, as $\hat{c}$ increases, the QAIC_c tends to increasingly favor models with fewer parameters.

* Some people feel that every model should have one additional parameter included if a value of c is estimated for the set of models. The is an option in **MARK** (under '**`File | Preferences`**') to automatically add 1 to K, the number of parameters estimated, for each model. However, the effect on model selection results is typically extremely small, and can result in errors in the value of K. Use at your own risk.

begin sidebar

What if $\hat{c}$ is < 1?

What if $\hat{c} < 1$? In the preceding, we mention the case where $\hat{c} > 1$, indicating some degree of lack of fit, reflecting (in all likelihood) overdispersion in the data. Now, if instead, $\hat{c} < 1$, then we generally consider this as reflecting underdispersion. While the intuitive thing to do is to simply enter the $\hat{c}$ as estimated (discussed below), there is lack of unanimity on how to handle $\hat{c} < 1$. Some authors recommend using the estimated $\hat{c}$, regardless of whether or not it is > 1 or < 1. However, still others suggest that if $\hat{c} < 1$, then you should set $\hat{c} = 1$ (i.e., make no adjustment to various metrics). For the moment, the jury is out - all we can recommend at this stage is - if $\hat{c} > 1$, then adjust. If $\hat{c} < 1$, then set $\hat{c} = 1$, and 'hold your nose'.

end sidebar

5.2. The practical problem - estimating $\hat{c}$

In a recent paper (*Journal of Applied Statistics*, **29**: 103-106), Gary White commented:

> *The Achilles' heel...in capture- recapture modeling is assessing goodness-of-fit (GOF). With the procedures presented by Burnham & Anderson (1998), quasi-likelihood approaches are used for model selection and for adjustments to the variance of the estimates to correct for over-dispersion of the capture-recapture data. An estimate of the over-dispersion parameter, c, is necessary to make these adjustments. However, no general, robust, procedures are currently available for estimating* c. *Although much of the goodness-of-fit literature concerns testing the hypothesis of lack of fit, I instead view the problem as estimation of c.*

So, the objective then becomes estimating the lack of fit of the model to our data. In other words, how to estimate $\hat{c}$? The general challenge of estimating $\hat{c}$ is the basis for a significant proportion of the remainder of this chapter.

As we'll see, there are a number of approaches that can be taken. The most obvious approach is to simply divide the model χ^2 statistic by the model degrees of freedom:

$$\hat{c} \cong \frac{\chi^2}{df}$$

However, in many (most?) cases involving the sorts of multinomial data we analyze with **MARK**, this approach doesn't work particularly well. Although the distribution of the deviance between the general and saturated models is supposed to be asymptotically χ^2 distributed, for the type of data we're working with it frequently (perhaps generally) isn't because of sparseness in the frequency table of some proportion of the possible encounter histories. For example, for live encounter mark-recapture data, for the CJS model (Chapter 4), there are $[(2^n - 1) - 1]$ possible encounter histories for n occasions, and for typical data sets, many of the possible encounter histories are either rare or not observed at all. The asymptotic distribution of the deviance assumes that all encounter histories are sampled (which would be true if the sample were infinitely large, which is of course the underlying assumption of 'asymptopia' in the first place). Given that the asymptotic assumptions are often (perhaps generally) violated for these sorts of data, alternative approaches are needed. Moreover, the χ^2 is not available for all models (in particular, for models where the saturated model is not defined), and there can be some non-trivial difficulties involved in the calculation of the χ^2 statistics, especially for sparse data sets. On the other

hand, the advantage of using a χ^2 approach is that the frequency tabulations used in deriving the χ^2 statistic are often very useful in determining the 'sources' of lack of fit.

In the following we'll discuss two broad categories of approaches for estimating $\hat{c}$. The first approach we'll describe, using program **RELEASE**, provides estimates of $\hat{c}$ for CJS live-encounter data using a contingency table (i.e., χ^2) approach. However, this is not generalizable to other data types, so other approaches are required.

The second approach we'll discuss (the bootstrap, and median $\hat{c}$) uses simulation and resampling to generate the estimate of $\hat{c}$. Rather than assuming that the distribution of the model deviance is in fact χ^2 distributed (since it generally isn't, for typical '**MARK** data'), the bootstrap and median $\hat{c}$ approaches generate the distribution of model deviances, given the data, and compare the observed value against this generated distribution. The disadvantage of the bootstrap and median $\hat{c}$ approaches (beyond some technical issues) is that both merely estimate $\hat{c}$. While this is useful (in a practical sense), it reveals nothing about the underlying sources of lack of fit.

Each approach has different strengths and weaknesses, so a good understanding of each of these procedures is important to successfully using **MARK**.

5.3. Program RELEASE - details, details. . .

For testing the fit of the data to a fully-time-dependent CJS model, program **RELEASE** has been the *de facto* standard approach for many years. In the following, we describe the use of **RELEASE** for GOF testing (and estimation of $\hat{c}$). We will discuss the use of **RELEASE** to generate specific GOF statistics, and give some broad suggestions for how to interpret lack-of-fit (from both a statistical and biological point of view), and what remedies are available. Note, **RELEASE** is primarily intended for standard live-recapture models, although it can be tweaked to handle some recovery models as well. While this may not be appropriate for your particular analysis (e.g., if you're working with telemetry, for example), there is still value in understanding how **RELEASE** works, since the principles underlying it are important for all analyses, not just standard live-recapture studies.

5.4. Program Release - TEST 2 & TEST 3

Program **RELEASE** generates 3 standard 'tests', which are given the absolutely uninformative names '**TEST 1**', '**TEST 2**', and '**TEST 3**'. The latter 2 tests, **TEST 2** and **TEST 3**, together provide the GOF statistics for the reference model (the time-dependent CJS model). **TEST 1** is an omnibus test that is generated **only** if you are comparing groups, and tests the following hypothesis under model $\{\varphi_{g*t}p_{g*t}\}$:

H$_o$: all parameters φ_i and p_i have the same value across treatment groups (i.e., there is no difference in survival (φ_i) or recapture (p_i) considered simultaneously among groups).

H$_a$: at least some values for either φ_i or p_i (or both) differ among groups.

The big advantage of using **MARK** or one of the other applications available for CMR analysis, is that you can separately model differences in either survival or recapture rate independently. **TEST 1** does not do this - it only tests for an 'overall' difference among groups. Since this severely limits its utility, we will not discuss use of **TEST 1** - in fact, we actively discourage it's use, since it is possible to do far more sophisticated analysis if you have capture histories from individually marked animals (although **TEST 1** may still be of use when complete capture histories are not available - see the 'blue book' for use of **RELEASE** and **TEST 1** under alternative capture protocols).

While **TEST 1** may be of limited use, **TEST 2** and **TEST 3** together are quite useful for testing the GOF of the standard time-dependent CJS (Cormack-Jolly-Seber) model to the data (this model was first presented in detail in Chapter 4). What do we mean by 'lack of fit'? As noted previously, we mean that the arrangement of the data do not meet the expectations determined by the assumptions underlying the model. These assumptions, which we also noted earlier in this chapter, sometimes known as the CJS assumptions are:

1. Every marked animal present in the population at time (i) has the same probability of recapture (p_i)
2. Every marked animal in the population immediately after time (i) has the same probability of surviving to time (i+1)
3. Marks are not lost or missed.
4. All samples are instantaneous, relative to the interval between occasion (i) and (i+1), and each release is made immediately after the sample.

For now, we will assume that assumptions 3 and 4 are met. It is assumptions 1 and 2 which are typically the most important in terms of GOF testing. In fact, **TEST 2** and **TEST 3** in **RELEASE**, as well as the GOF tests in other software, directly test for violations of these two assumptions (in one form or another).

Let's expand somewhat on assumptions 1 and 2. Assumption 1 says that all marked animals in the population have the same chances of being captured at any time (i). What would be the basis for violating this assumption? Well, suppose that animals of a particular age or size are more (or less) likely to be captured than animals of a different age or size? Or, suppose that animals which go through the process of being captured at occasion (i) are more (or less) likely to be captured on a later occasion than animals who were marked at some other occasion? Or, what if some marked individuals temporarily leave the sample area (temporary emigration)? Or what if animals always exist in 'pairs'? For estimation of survival in open populations, marked animals have the same probability of recapture. For estimation of population size (abundance), both marked and unmarked animals must have the same probability of capture.

What about assumption 2? Assumption 2 says that, among the marked individuals in the population, all animals have the same probability of surviving, regardless of when they were marked. In other words, animals marked at occasion (i-1) have the same probability of surviving from (i) to (i+1) as do animals marked on occasion (i). When might this assumption be violated? One possibility is that individuals caught early in a study are more (or less) prone to mortality than individuals caught later in the study. Or, perhaps you are marking young individuals. An individual captured and marked as an offspring at (i-1) will be older, or larger, or possibly of a different breeding status, at occasion (i), while offspring marked at occasion (i) are just that, offspring. As such, the offspring marked at (i-1) may show different survival from (i) to (i+1) than offspring marked at (i), since the former individuals are older, or larger, or somehow 'different' from individuals marked at (i).

For both **TEST 2** and **TEST 3** we have noted several reasons why either **TEST 2** or **TEST 3** might be violated. The examples noted are by no means an all-inclusive list - there are many other ways in which either or both tests could be violated. While violation of the underlying model assumptions has a specific statistical consequence (which we will deal with shortly), it may also serve to point out something interesting biologically. For example, suppose all animals are not equally likely to be captured at any occasion. We might ask 'why? Does this reveal something interesting about the biology?'.

We'll approach GOF testing in 2 steps. First, we'll describe the 'mechanics' of how to run **RELEASE** to generate the **TEST 2** and **TEST 3** results. Then, we'll discuss the mechanics of how these two tests are constructed, and how to interpret them.

5.4.1. Running RELEASE

Running **RELEASE** from within **MARK** is easy. Running it as a standalone application is also fairly straightforward - more on this in a moment. For now, we will restrict our discussion to running **RELEASE** from within **MARK**, although there may be a few instances where it may become necessary to run **RELEASE** as a stand-alone application.

To run **RELEASE** from within **MARK**, simply pull down the '**Tests**' menu, and select '**Program RELEASE GOF**'. This option is available only if you selected '**Recaptures**' as the data type. That's it. **RELEASE** will run, and the results will be output into a Notepad window.

At the top of this output file there will be some information concerning recent updates to the **RELEASE** program, and some statement concerning limits to the program (maximum number of groups, or occasions). Then, you will see a listing of the individual capture histories in your data set, plus a summary tabulation of these histories known as the *reduced m-array*. The *m*-array contains summary information concerning numbers of individuals released at each occasion, and when (and how many) of them were captured at subsequent occasions. The reduced *m*-array will be discussed in more detail later. These *m*-array tabulations are then followed by the **TEST 2** and **TEST 3** results for each group (respectively), followed in turn by the summary statistics.

TEST 2

TEST 2 deals only with those animals known to be alive between (i) and (i+1). This means we need individuals marked at or before occasion (i), and individuals captured at or later than (i+1). If they were alive at (i), and captured at or later than (i+1), then they must have been alive in the interval from occasion (i) to (i+1).

In other words, 'is the probability of being seen at occasion (i+1) a function of whether or not you were seen at occasion (i), given that you survived from (i) to (i+1)?'. Under assumption 1 of the CJS assumptions, all marked animals should be equally 'detectable' at occasion (i+1) independent of whether or not they were captured at occasion (i). **TEST2.C** has the following general form: of those marked individuals surviving from (i) to (i+1), some were seen at (i+1), while some were seen after (i+1). Of those not seen at (i+1), but seen later, does 'when' they were seen differ as a function of whether or not they were captured at occasion (i)?

In other words:

	when seen again?					
seen at (i)	(i+1)	(i+2)	(i+3)	(i+4)	...	(i+5)
no	f	f	f	f	f	f
yes	f	f	f	f	f	f

So, **TEST2** asks 'of those marked animals not seen at (i+1), but known to be alive at (i+1) (since they were captured after i+1), does when they were next seen (i+2, i+3...) depend on whether or not they were seen at (i)?'. Again, we see that **TEST2.C** deals with capture heterogeneity. For most data sets, pooling results in a (2×2) matrix.

TEST2 (in general) is sensitive to short-term capture effects, or non-random temporary emigration. It highlights failure of the homogeneity assumption (assumption 1), among animals and between occasions. In practice, **TEST 2** is perhaps most useful for testing the basic assumption of 'equal catchability' of marked animals. In other words, we might loosely refer to **TEST 2** as the 'recapture test'.

TEST 3

In general, **TEST 3** tests the assumption that all marked animals alive at (*i*) have the same probability of surviving to (*i*+1) - the second CJS assumption.

TEST 3 asks: 'of those individuals seen at occasion (*i*), how many were ever seen again, and when?'. Some of the individuals seen at occasion (*i*) were seen for the first time at that occasion, while others had been previously seen (marked). Does whether or not they were ever seen again depend on this conditioning? The first part of **TEST 3**, known as **TEST3.SR**, is shown in the following contingency table:

seen before (*i*)	**seen again**	**not seen again**
no	*f*	*f*
yes	*f*	*f*

In other words, does the probability that an individual known to be alive at occasion (*i*) is ever seen again depend on whether it was marked at or before occasion (*i*)? If there is only a single release cohort, then 'seen before *i*?' becomes 'seen before *i*, excluding initial release?'.

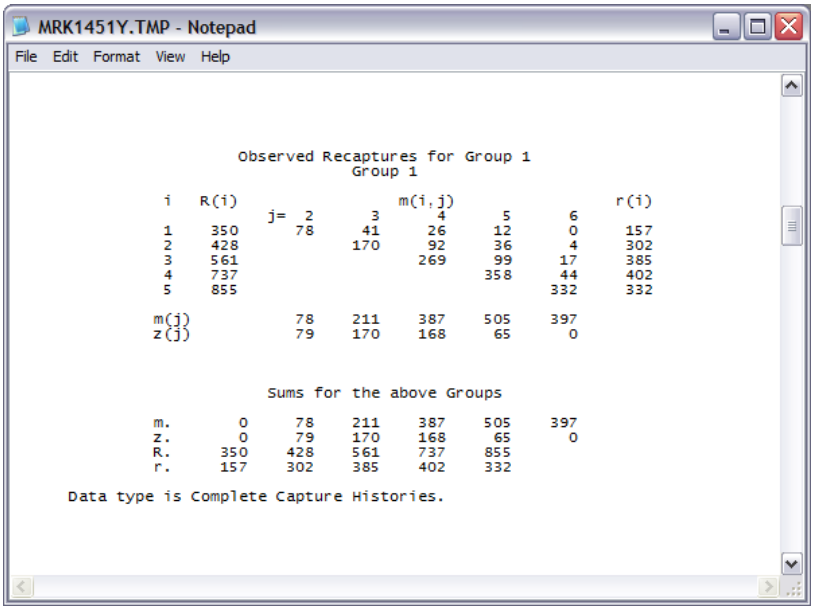
TEST3.SR is what is presented for **TEST 3** in the version of **RELEASE** bundled with **MARK**. There is also a **TEST3.Sm**, which asks '...among those animals seen again, does **when** they were seen depend on whether they were marked on or before occasion (*i*)?'. **TEST3.Sm** is depicted in the following contingency table:

	when seen again?					
seen before (*i*)	(*i*+1)	(*i*+2)	(*i*+3)	(*i*+4)	...	(*i*+5)
no	*f*	*f*	*f*	*f*	*f*	*f*
yes	*f*	*f*	*f*	*f*	*f*	*f*

If there is only a single release cohort, then 'seen before *i*?' become 'seen before *i*, excluding initial release?'. So, in a very loose sense, **TEST 2** deals with 'recapture problems', while **TEST 3** deals with 'survival problems' (although there is no formal reason to make this distinction - it is motivated by our practical experience using **RELEASE**). If you think about it, these tables should make some intuitive sense: if assumptions 1 and 2 are met, then there should be no difference among individuals if or when they were next seen conditional on whether or not they were seen on or before occasion (*i*).

Let's consider a simple example of GOF testing with **RELEASE**. We simulated a small data set - 6 occasions, 350 newly marked individuals released alive at each occasion. First, let's look at something call the *reduced m-array* table **RELEASE** generates as the default (the other *m*-array presentation you can generate running **RELEASE** as a stand-alone application is the *full m-array* - this will be discussed later). Examination of the *m*-array will give you some idea as to 'where the numbers come from' in the **TEST 2** and **TEST 3** contingency tables.

Here is the reduced *m*-array:


MRK1451Y.TMP - Notepad

File Edit Format View Help

Observed Recaptures for Group 1
Group 1

i	R(i)	m(i,j)				r(i)
		j= 2	3	4	5	6	
1	350	78	41	26	12	0	157
2	428		170	92	36	4	302
3	561			269	99	17	385
4	737				358	44	402
5	855					332	332
m(j)		78	211	387	505	397	
z(j)		79	170	168	65	0	

Sums for the above Groups

m.	0	78	211	387	505	397
z.	0	79	170	168	65	0
R.	350	428	561	737	855	
r.	157	302	385	402	332	

Data type is Complete Capture Histories.

The main elements of interest are the $\mathbf{R}_i$, $m_{i,j}$, and r_i values. The $\mathbf{R}_i$ values are the number of individuals in total released on each occasion. For example, $\mathbf{R}_1 = 350$ equals 350 individuals released on the first occasion - all newly marked. At the second occasion ($\mathbf{R}_2$), we released a total of 428 individuals - 350 newly marked individuals, plus 78 individuals from the first release which were captured alive at the second occasion. The $m_{i,j}$ values are the number of individuals from a given release event which are captured for the first time at a particular occasion. For example, $m_{1,2} = 78$. In other words, 78 of the original 350 individuals marked and released at occasion 1 (i.e., $\mathbf{R}_1$) were recaptured for the first time at occasion 2. At the third occasion ($m_{1,3}$), 41 individuals marked and released at occasion 1 were recaptured for the first time, and so on.

The r_i values are the total number of individuals captured from a given release batch (see below). For example, from the original $\mathbf{R}_1 = 350$ individuals, a total of 157 were recaptured over the next 5 capture occasions. Neither the $m_{i,j}$, or r_i values distinguish between newly marked or re-released individuals - they are simply subtotals of all the individuals released at a given occasion. As we'll see shortly, this limits the usefulness of the reduced *m*-array.

begin sidebar

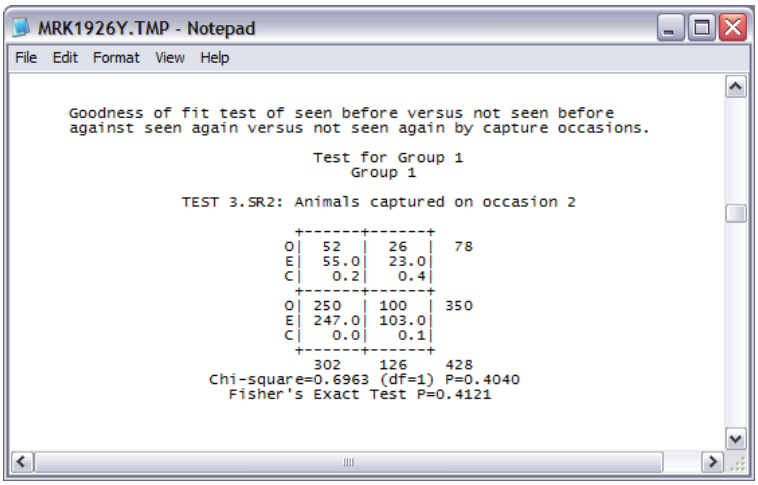
Batch release??

What do we mean by a 'batch release'? Historically, a *cohort* referred to a group of animals released at the same occasion - whether newly marked or not. However, when using **MARK**, we refer to a cohort as all animals marked at the some occasion. In this context, an animal does not change cohorts - it is a 'fixed' characteristic of each marked individual. In the **RELEASE** context, cohort changes with each capture occasion. To prevent confusion, we use the term 'release batch', or simply 'batch', to refer to all individuals (marked and unmarked) released on a given occasion.

end sidebar

Following the reduced *m*-array are the results for **TEST 3**. Since there are 5 recapture occasions there are as many as 7 total **TEST 3** tables (4 for **TEST3.SR** and 3 for **TEST3.Sm**). Let's consider just one of

these tables - the first **TEST3.SR** table, for individuals captured on occasion 2.



Why is this the first table? Well, recall what **TEST3.SR** compares - seen before versus not seen before - obviously, at occasion 1, **no** animals were seen before. Thus, we start at occasion 2.

Look closely at the table. Note that the table starts with a 'loose' restatement of what is being tabulated - in this case 'goodness of fit test of seen before versus not seen before against seen again versus not seen again by capture occasions'. You will also see comments concerning which group is being tested, and possibly something concerning the 'control' group. By default, if you're working with only one group, **RELEASE** assumes that it is a 'control group' in a 'control vs. treatments' experiments. Then, comes the contingency table itself. First, note the labeling: **TEST3.SR**2. The '**TEST.3SR**' part is obvious, the '2' simply refers to the second occasion (so, **TEST3.SR3** for the third occasion, **TEST3.SR4** for the fourth occasion, and so on). At occasion 2, a total of 428 individuals were released. Of these, 78 had been seen before, and 350 were newly marked individuals. In the first row of the contingency table, we see that of the 78 individuals seen before, a total of 52 (or 67%) of these individuals were ever seen again. In the second row of the table, we see that of the 350 newly marked individuals, a total of 250 (or 71%) were ever seen again. Where did the numbers 52 and 250 come from? Can we tell from the reduced *m*-array? Unfortunately, the answer is 'no'. Why? Because the reduced *m*-array does not 'keep track' of the fates of individuals depending on when they were marked. For this, you need a different type of *m*-array, known as the *full m*-array. To generate the full *m*-array, you need to run **RELEASE** as a standalone application, and modify a specific control element to generate the full *m*-array.

5.4.2. Running RELEASE as a standalone application

To run **RELEASE** as a stand-alone application, you first need to make a simple modification to the `INP` file containing the encounter history data. You simply need to add a single line to the top of the `INP` file. The 'additional' line is the `PROC CHMATRIX` statement. Here is the minimal `PROC CHMATRIX` statement for our example data set:

```
PROC CHMATRIX OCCASIONS=6 GROUPS=1;
```

The `PROC CHMATRIX` statement must include (at least) the `GROUPS` and `OCCASIONS` statements. However, there are a number of other options which can also be applied to this procedure. One of these options is

DETAIL - as its name implies, the DETAIL option provides 'detailed' information about something. The DETAIL option is the default in the version of **RELEASE** which comes with **MARK**. The 'something' is in fact detailed information concerning **TEST 2** and **TEST 3**. When the DETAIL option is in effect, **RELEASE** provides the individual contingency tables (including observed and expected frequencies) upon which they are based (discussed below), as well as the summary statistics for all batches pooled. If you have a data set with a large number of occasions, this can generate a very large amount of output.

The opposite to the DETAIL option is the SUMMARY option, which forces **RELEASE** to print only the summary **TEST 2** and **TEST 3** results for each batch separately and all batches pooled.

You choose either the DETAIL or SUMMARY option as follows:

```
PROC CHMATRIX OCCASIONS=6 GROUPS=1 DETAIL;
```

To use the SUMMARY option (instead of DETAIL), you would type

```
PROC CHMATRIX OCCASIONS=6 GROUPS=1 SUMMARY;
```

To generate a full *m*-array (below) you would simply write:

```
PROC CHMATRIX OCCASIONS=6 GROUPS=1 DETAIL FULLM;
```

How do you run **RELEASE**? Simply shell out to DOS, and type:

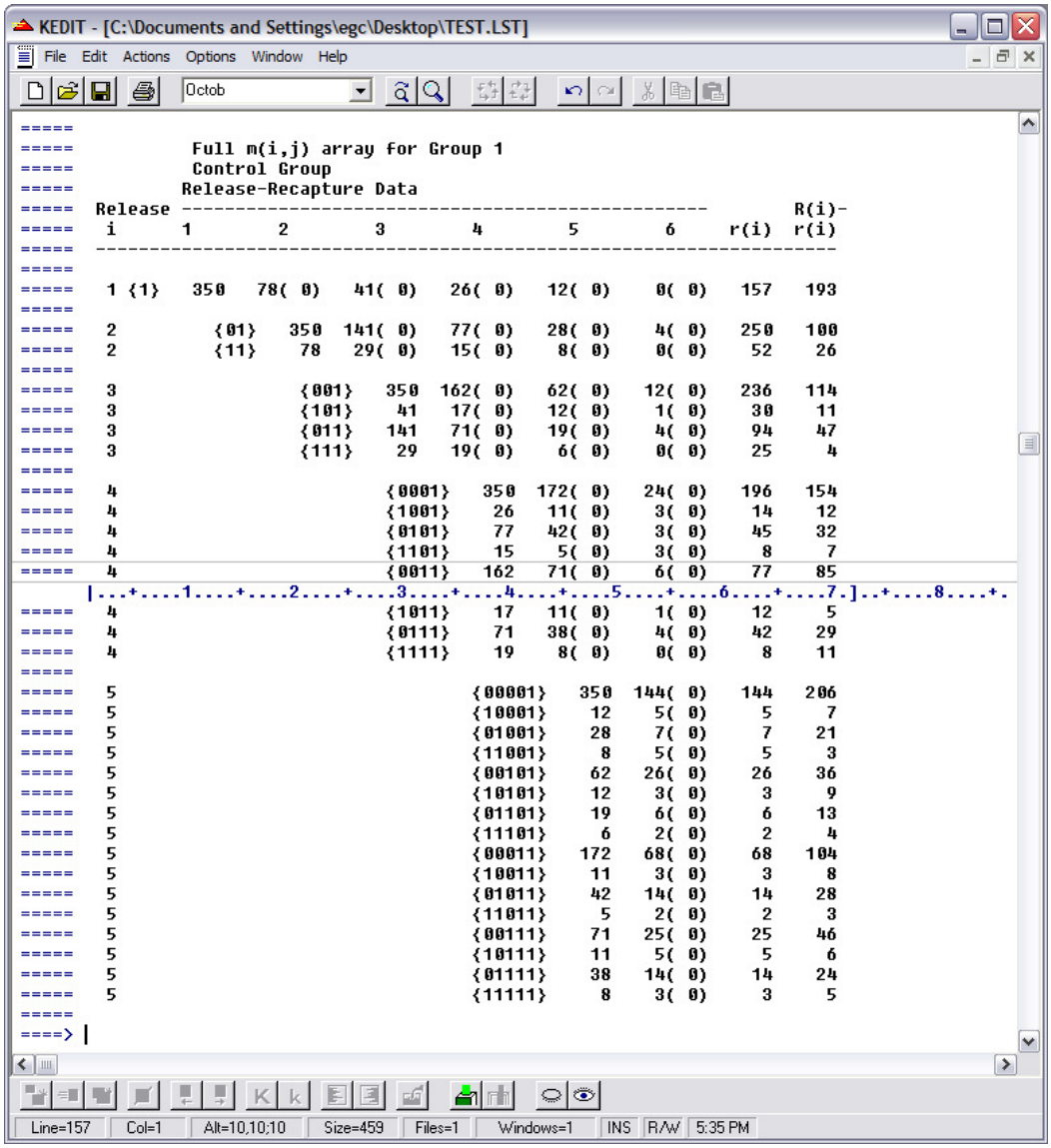
```
REL_32 I=<INPUT FILE> O=<OUTPUT FILE> <enter>
```

If nothing happens, it probably means that **REL_32** isn't in the PATH on your computer. Make sure it is, and try again. If our **RELEASE** file is called TEST.REL, and we want our results to be written to a file called TEST.LST, then we would type:

```
REL_32 I=TEST.REL O=TEST.LST  <enter>
```

The output would be in file TEST.LST, which you could examine using your favorite text editor. Now, for the present, we're interested in considering the full *m*-array. Assume that we've successfully added the appropriate PROC CHMATRIX statement to the INP file for our simulated data, and successfully run **RELEASE**. In the output, we see something that looks quite different than the simple, reduced *m*-array. This is the full *m*-array, and is shown at the top of our next page for our example, simulated data.

As you can readily see, the full *m*-array contains **much** more information than the reduced m-array. In fact, it contains the entire data set!! If you have the full *m*-array, you have all the information you need to run a CMR analysis. If you look closely at the full *m*-array, you'll see why. Let's concentrate on the information needed to generate **TEST3.SR2**. From the preceding page, recall that of the 78 individuals (i) marked at occasion 1, that (ii) were also captured and re-released at occasion 2, 52 were seen again at some later occasion. What would the capture history of these 78 individuals be? - obviously '11' - marked at the first occasion, and recaptured at the second occasion. The '11' capture history is represented as {11} in the full *m*-array. Find this capture history in the 3rd line. To the right, you will see the number 78, indicating that there were 78 such individuals. To the right of this value are the totals, by capture occasion, of individuals from this group of 78 ever seen again. For example, 29 of this 78 were seen again for the first time at occasion 3, 15 were seen for the first time at occasion 4, and so on. In total, of the 78 {11} individuals released, a total of 52 were seen again. You should now be able to see where the values in the **TEST3.SR2** table came from.



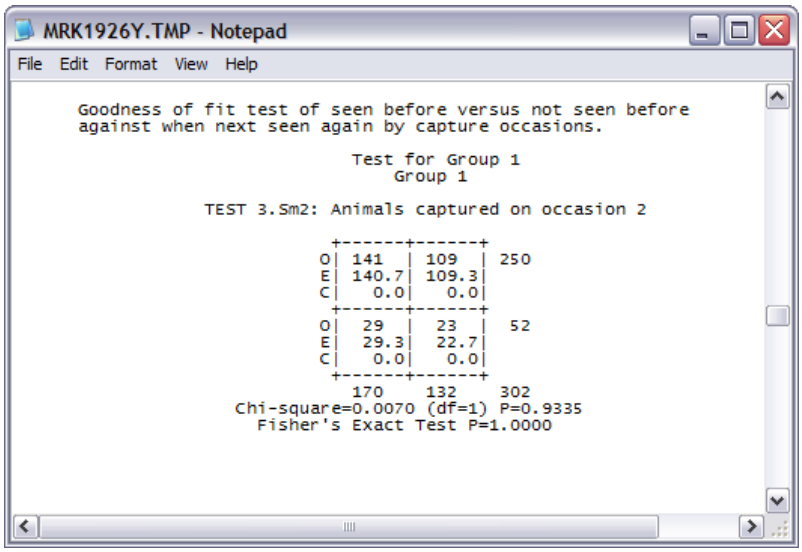
Now, consider the 'statistical results'. Although the proportions seen again appear to differ between the two groups (68% for previously marked vs 71% for the newly marked), they are not statistically different ($\chi^2_1 = 0.696$, P=0.412). What are the other 2 numbers in each of the cells? Well, if you look down the left side of the table you'll get a hint - note the 3 letters 'O', 'E' and 'C'. 'O' = the observed frequencies, 'E' = the expected frequencies (under the null hypothesis of the test), and 'C' represents the contribution to the overall table χ^2 value (summing the 'C' values for all four cells yield 0.696. The 'C' values are simply $(O-E)^2/E$). So, for individuals released at the second occasion, there is no significant difference in 'survival' between newly marked and previously marked individuals.

Following the last table (**TEST3.SR5** - individuals released at occasion 5), **RELEASE** prints a simple cumulative result for **TEST3.SR** - which is simply the sum of the individual χ^2 values for each of the individual **TEST3.SRn** results. In this case, $\chi^2 = 2.41$, df=3, $P = 0.492$. What if **TEST.3SR** had been significant? As we will see shortly, examination of the individual tables is essential to determine the possible cause of lack of fit. In this case, since we have no good 'biological explanation' for **TEST3.SR3**

(obviously, since this is a simulated data set!), we accept the general lack of significance of the other tables, and conclude that there is no evidence over all occasions that 'survival' differs between newly marked and previously marked individuals.

Now let's look at **TEST3.Sm2** (i.e., **TEST3.Sm** for occasion 2). Recall that this test focuses on 'of those individuals seen again, when were they seen again, and does when they were seen differ among previously and newly marked individuals?'. As with **TEST3.SR**, there is a contingency table for each of the batches, starting with the second occasion, and ending with occasion 4.

Why not occasion 5? Well, think about what **TEST3.Sm** is doing - it is comparing when individuals are seen again (as opposed to are they seen again). At occasion 5, any individual if seen again must have been seen again at the last occasion (6), since there are no other occasions! So, it doesn't make much sense to create **TEST3.Sm** for the penultimate occasion. Let's consider **TEST3.Sm2** - the second occasion.



At occasion 2, a total of 428 individuals were released - 78 that had been seen before, and 350 newly marked individuals. Of these 428 individuals, 302 were seen again. From the **TEST3.Sm2** table (above), 250 of this 302 were newly marked individuals, and 52 were previously marked. You should be able to determine where these totals come from, using the full *m*-array (shown on the preceding page).

However, we're now faced with a different puzzle - why only two columns? If **TEST3.Sm** considers 'when' individuals were seen again, then unless all individuals seen again were seen on only the next two occasions, then there should be more than two columns.

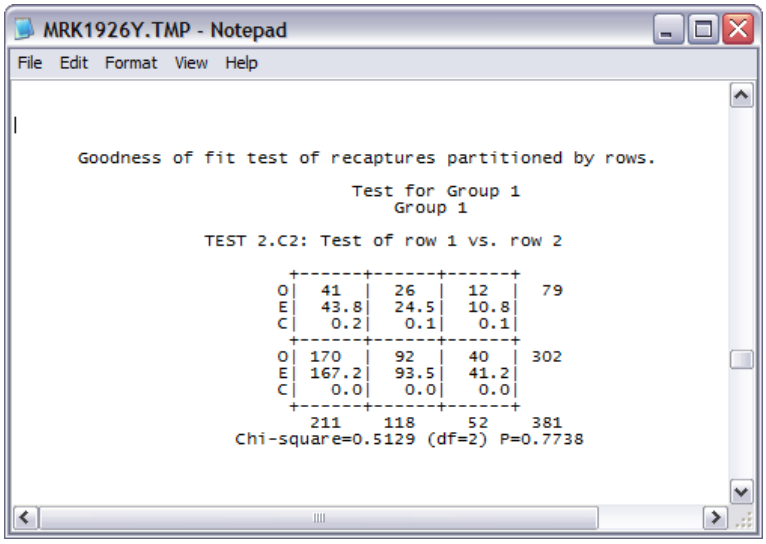
Look at the full *m*-array (on the preceding page). We see that of the 428 individuals marked and released at occasion 2, 350 were newly marked and released (the {01} individuals), while 78 were previously marked at occasion 1, and released (the {11} individuals). Of the 350 {01} individuals, 141 were seen again for the first time at occasion 3, 77 were seen again for the first time at occasion 4, and so on. Among the 78 {01} individuals, 29 were seen again for the first time at occasion 3, 15 were seen again for the first time at occasion 4, and so on.

Thus, if we were to construct our own **TEST3.Sm2** table, it would look like:

TEST3.Sm2	when seen again?			
seen at (2)	(3)	(4)	(5)	(6)
{01}	141	77	28	4
{11}	29	15	8	0

So why doesn't the **RELEASE** table for **TEST3.Sm2** look like this? It doesn't, because **RELEASE** is 'smart' enough to look at the 'true' table (above) and realize that the data are too sparsely distributed for the later occasions for a contingency test to be meaningful. **RELEASE** has simply pooled cells, collapsing the (2×4) table to a (2×2) table.

Now consider **TEST 2**, starting with **TEST2.C**. Recall that in **TEST2.C**, we are 'using' individuals that are known to have survived from (i) to (i+1). **TEST2.Ct** tests if the probability of being seen at occasion (i+1) is a function of whether or not the individual was seen at occasion (i), conditional on surviving from (i) to (i+1). **TEST 2** differs from **TEST 3** somewhat in that we are not considering when an individual was marked, but rather on when it was recaptured. The result for **TEST.2C2** is shown below:



Once each of the component tests **TEST 3** and **TEST 2** are finished, **RELEASE** presents you with a convenient tabulation of all of the individual **TEST 3** and **TEST 2** results. It also gives you some indication as to whether or not there was sufficient data in a given test for you to be able to 'trust' the result. Using our simulated data, we have no significant **TEST 2** or **TEST 3** result. Thus, the overall GOF result (**TEST 2** + **TEST 3** = 6.34) is also not significant (P = 0.898). This is perhaps not surprising, since we set up the simulation so that the data **would** follow the CJS assumptions! Our purpose here was simply to introduce **TEST 2** and **TEST 3**.

One thing you might be asking yourself at this point is 'since **RELEASE** gives me these nice summary tables, why do I need so much detail?'. The answer – if your data *do* fit the CJS model, then you clearly don't. But if your data don't fit the model (i.e., if any of the tests is rejected), then the only chance you have of trying to figure out what is going on is to look at the individual contingency tables. We got some sense of this when we looked at **TEST3.SR** in our simulated data set - one of the batches had results

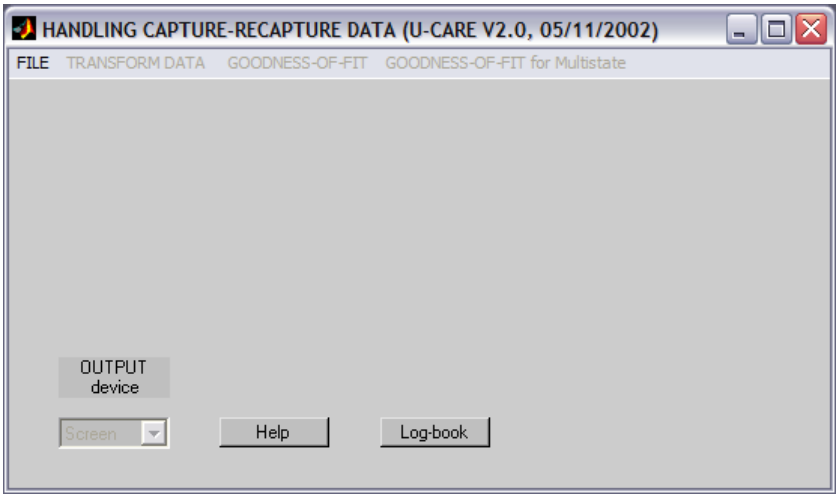
quite different from the other batches, leading to a near-significant **TEST3.SR** result overall. Further, even if the 4 tests are accepted (no significant differences) you should remember that these tests are for simple heterogeneity - they do not test specifically for systematic differences. Again, the only clue you might have is by looking at the individual tables.

5.5. Enhancements to RELEASE - program U-CARE

Recently, Rémi Choquet, Roger Pradel, and Olivier Gimenez have developed a program (known as **U-CARE**, for **U**nified **Ca**pture-**Re**capture) which provides several enhancements to program **RELEASE**. In its previous versions, **U-CARE** provided goodness-of-fit tests for single-site models only. Recently, new tests have appeared for the multistate **J**olly**M**o**V**e (JMV) and Arnason-Schwarz (AS) models (Pradel, R., C. Wintrebert and O. Gimenez, 2003) and those tests have been incorporated in the current version of **U-CARE** (for discussion of the use of **U-CARE** for GOF testing for multi-state models, see the last section(s) of Chapter 8). Here, we concentrate on using **U-CARE** for GOF testing for single-state models only.

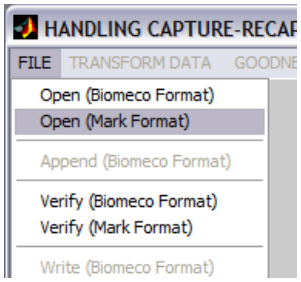
U-CARE contains several tests which are similar to those found in **RELEASE**, but in many cases using slightly different strategies for pooling sparse contingency tables (and thus, the results may differ slightly from those from **RELEASE** - we'll see an example of this shortly). More importantly, however, **U-CARE** incorporates specific 'directional' tests for transience (Pradel *et al.*, 1997) and trap-dependence (trap-happiness or trap shyness; Pradel 1993) derived from the contingency tables used in the GOF tests in **RELEASE**. Forthcoming versions of **U-CARE** are anticipated to incorporate further specialized tests and appropriate treatment of sparse data together with indications on the recommended model from which model selection can start.

At present, **U-CARE** cannot be run from within **MARK**, and so must be run separately, as a stand-alone program. When you start **U-CARE**, you will be presented with two windows : one, a 'black DOS window' (which is where evidence of the numerical estimations can be seen - you may have already noticed that **MARK** uses a similar 'command window' during its numerical estimations), and the main 'graphical' front-end to **U-CARE** – clearly, it's pretty 'minimalist':



Initially, only one menu is available: the 'File' menu. As you might expect, this is where you tell **U-CARE** where to find the data you want to perform a GOF test on.

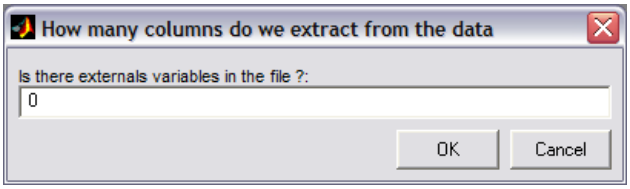
However, if you access the '**File**' menu,



you will see a number of options: you can open encounter history files in one of two formats: a format used in program **SURGE** (and derivatives) known as **Biomeco**, and the one we're familiar with, the **MARK** format (the distinctions between the formats are minor - in fact, **U-CARE** provides you the utility function of being able to read in data in one format, verify it, and write it out in another format.

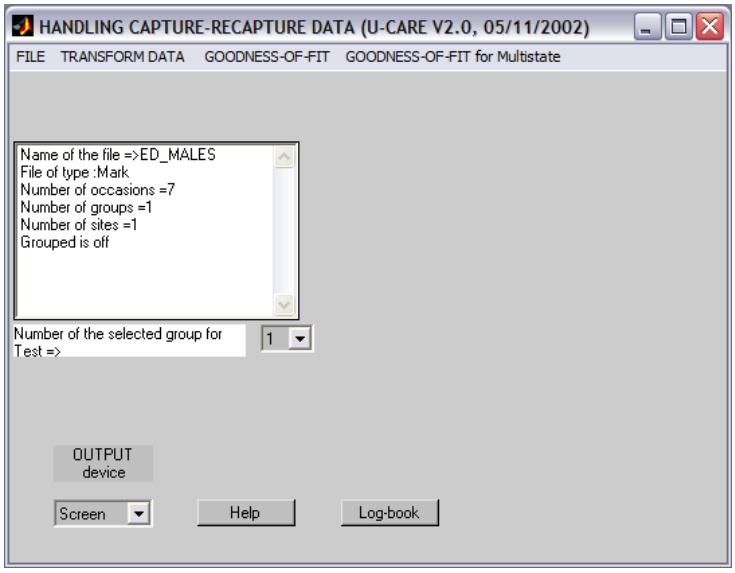
To demonstrate using **U-CARE**, let's test the fit of a familiar data set - the European dippers. We'll focus for the moment on the males only (i.e., a single group). This file is called `ed_males.inp`. We simply select this file using the '**Open (MARK Format)**' file option in **U-CARE**.

Once you've selected the **MARK** input file, **U-CARE** responds with a small 'pop-up' window which is asking you (in effect) if there are any external covariates in the file (see Chapter 2). In this case, with the male Dipper data, there are no covariates included in the data set, so **U-CARE** informs you that, as far as it can tell, there are 0 covariates:



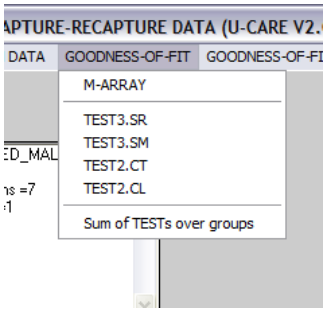
If this is correct (which it is in this case), simply click the '**OK**' button to proceed. You'll then be presented with 2 new windows: one window shows the encounter histories themselves (if they look 'strange' - specifically, if you're wondering why the columns are separated by spaces - not to worry. This is `Biomeco` format, and is functionally equivalent to **MARK**'s input format in how **U-CARE** processes the data).

The other window (shown at the top of the next page) is the main **U-CARE** window, but with many more options now available, plus a window showing you some details about the file you just read in. Note that **U-CARE** assumes that the number of occasions in the data file is the number of occasions you want to test GOF over. In **MARK**, recall that you must 'tell **MARK**' how many occasions there are (or, that you want to use).



If you pull down each of the menus in turn, you'll see that there are a **lot** of options in **U-CARE**. The '**`Transform Data`**' menu provides a set of convenient ways in which to split or pool data (e.g., pooling multiple strata into a single stratum), according to various criterion, reverse the encounter histories, and so forth.

The other two menu options are clearly relevant for GOF testing. There is a GOF menu, and then one specific to multi-state models. For the moment, since our example data have only one 'state' (multi-state models is something we cover is some detail in Chapter 8), we'll consider only the '**`Goodness-of-Fit`**' menu. If you access this menu, you'll see several options.

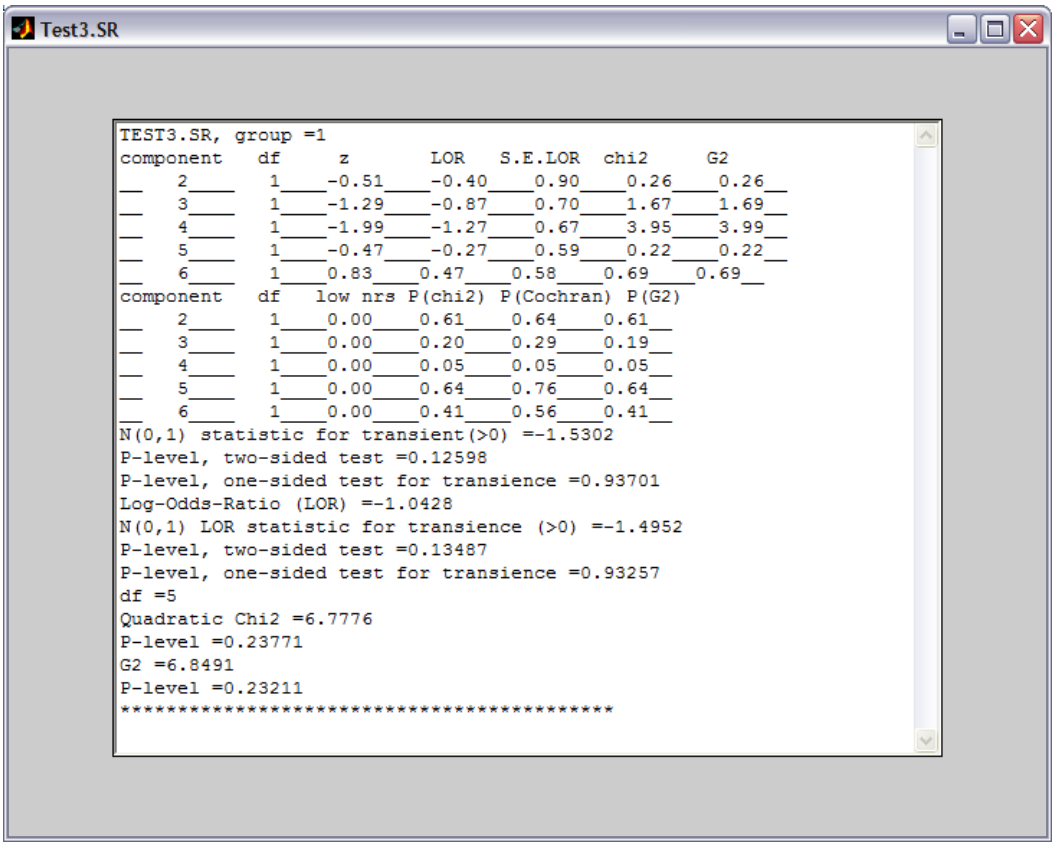


The first ('**`M-ARRAY`**') allows you to generate the reduced *m*-array for your data. Recall that the reduced *m*-array is a summary table, and does not represent all of the details concerning the encounter histories, which are contained in the full *m*-array. The *m*-array is useful for 'visual diagnosis' of some aspects of your data.

Next are 4 component tests: two for '**Test 3**', and two for '**Test 2**'. The use of '**Test 3**' and '**Test 2**' indicates clearly that **U-CARE** is built on the underlying principles (and code base) of program **RELEASE**. In some cases, the tests are identical (for example, **TEST3.SR**). In other cases, they are somewhat different (e.g., there is no **TEST2.CL** in the version of **RELEASE** distributed with **MARK**). More on these individual component tests in a moment. Finally, there is an option (at the bottom of the menu) to sum the tests over groups. This option basically gives you the summary results of the individual

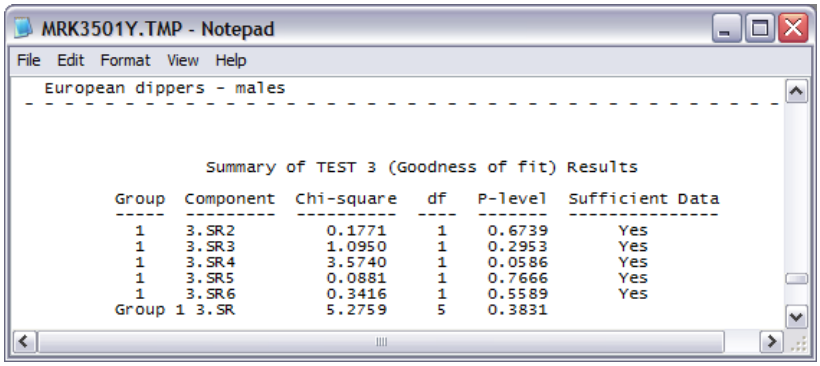
component tests, in a single output.

To explore the individual component tests in **U-CARE**, let's proceed to do the GOF testing on the male dippers. We'll start with **TEST3.SR**. Recall from the earlier discussion of program **RELEASE** that **TEST3.SR** tests the hypothesis that there is no difference among previously and newly marked individuals captured at time (i) in the probability of being recaptured at some later time $> i$ (i.e., that whether or not an animal is ever encountered again is not a function of whether or not it is newly marked). If you select **TEST3.SR** from the menu, **U-CARE** will respond with a window showing the contributions of each cohort to the overall χ^2 statistic for this test:



One of the first things we notice from the output for **TEST3.SR** (and all the other tests, which will get to shortly) is that **U-CARE** provides a fair number more 'statistical bits' than you find in the output from program **RELEASE**. For example, you'll recall from our preceding discussion of program **RELEASE** that by careful examination of the individual contingency tables of **TEST3.SR**, you might be able to 'visually' detect systematic departures from expectation in the contingency tables, which might be consistent with transient effects (or age effects). However, **U-CARE** formalizes this level of analysis (while also making it much simpler), by providing a test specific for 'transience' (or, perhaps more accurately, directionality). In fact, **U-CARE** gives you 2 different approaches to this statistic (the second one based on a log-odds-ratio), as well as both a two-tailed and one-tailed significance test. **U-CARE** also provides two test statistics for overall heterogeneity (the quadratic and likelihood-based G test). The table-like material at the top of the output is the contribution of each cohort to the various statistics (the additivity of the various statistics is quite useful, since it can help you identify particular cohorts which might be having undue influence on the overall fit).

How do these results compare to those from **RELEASE**? Recall we mentioned in passing that **U-CARE** uses a slightly different pooling algorithm than does **RELEASE**, and as such, there will be occasions where **U-CARE** and **RELEASE** give slightly different answers. Here are the results from **TEST3.SR** from program **RELEASE**.



We see that the overall heterogeneity χ^2 statistic from **RELEASE** (which is based on a Pearson statistic) is 5.2759, with 5 df. Based on a two-tailed test, the calculated probability is 0.3831. From **U-CARE**, there are two test statistics: 6.7776 and 6.8491, both with the same degree of freedom (5). Both of these values are somewhat higher than the value from **RELEASE**. These differences come from differences in how pooling in sparse cohort-specific contingency tables is handled between the two programs. You can get a sense of this by comparing the contributions from each cohort to the overall χ^2 statistic between the two programs. Note that the differences are quite striking in this example: many of the cohorts have very sparse data.

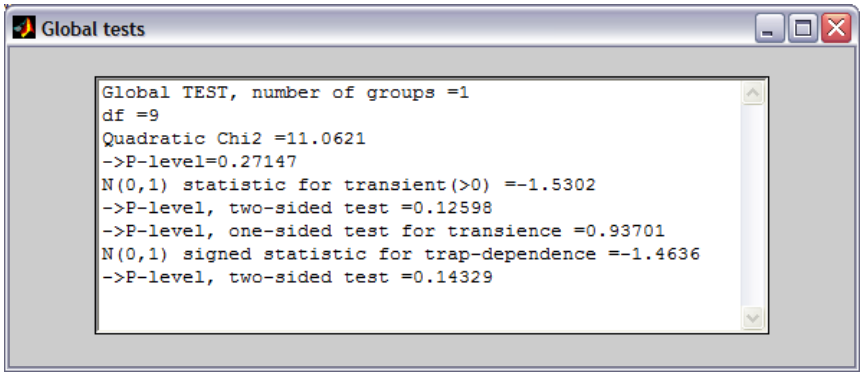
What about the other component tests? In **RELEASE**, there is a companion test for **TEST3.SR**, referred to as **TEST3.Sm** (recall that **TEST3.Sm** tests the hypothesis that there is no difference in the expected time of first recapture between the 'new' and 'old' individuals captured at occasion *i* and seen again at least once). This test is also available in **U-CARE**.

However, there are some notable differences between **MARK** and **U-CARE** when it comes to **TEST2**. In **MARK**, there is only a single **TEST2** provided (**TEST2.C**), whereas in **U-CARE**, **TEST2** is divided into two component tests: **TEST2.CT**, and **TEST2.CL**. **TEST2.CT** tests the hypothesis that there is no difference in the probability of being recaptured at *i*+1 between those captured and not captured at occasion *i*, conditional on presence at both occasions. **TEST2** differs from **TEST3** somewhat in that we are not considering when an individual was marked, but rather on when it was recaptured. The **TEST2.C** in **MARK** is equivalent to **TEST2.CT** in **U-CARE**. But, what about **TEST2.CL**, which is presented in **U-CARE**? **TEST2.CL**, based on the contingency table where individuals are classified on whether or not they were captured before (or at) occasion *i*, and after (or at) occasion *i*+2 (and thus known to be alive at both *i*, and *i*+1). The null hypothesis being tested in **TEST2.CL** is that there is no difference in the expected time of next recapture between the individuals captured and not captured at occasion *i* conditional on presence at both occasions *i* and *i*+2. To date, this test has no 'simple' interpretation, but it is a component test of the overall **TEST2** fit statistic.

5.5.1. RELEASE & U-CARE – estimating $\hat{c}$

OK, so now we have several **TEST3** and **TEST2** component statistics. What do we need these for? Well, clearly one of our major motivations is assessing fit, and (more mechanically) deriving an estimate of

the $\hat{c}$ value we'll use to adjust for lack of fit. Using either **U-CARE**, or **RELEASE**, one estimate for $\hat{c}$ is to take the overall χ^2 (sum of the **TEST 2** and **TEST 3** component tests), and divide by the overall degrees of freedom. If we use **RELEASE**, we see that the overall **TEST 3** statistic is 5.276 (for **TEST3.SR**), and 0.000 (for **TEST3.SM**), for an overall **TEST 3** χ^2 = 5.276. For **TEST 2**, there is only one value reported in **RELEASE**: **TEST2.CT** χ^2 = 4.284. Thus, the overall model χ^2 = (5.276 + 4.284) = 9.56, with 9 df. The probability of a χ^2 value this large, with 9 df, is reported as 0.3873. Thus, the estimate for $\hat{c}$, based on the results from **RELEASE**, is (9.56/9) = 1.06, which is close to 1.0. From **U-CARE**, we can get the 'overall' statistics quite easily, simply by selecting '**`Sum of tests over groups`**'. When we do so, we get the following output:



The overall test statistic is reported as 11.0621, with 9 df, yielding an estimate of $\hat{c}$ = (11.062/9) = 1.23, which is somewhat larger than the value calculated from the **RELEASE** output. But, note that **U-CARE** also provides some further diagnostics: specifically, tests of transience, and trap-dependence. In this case, for the male dipper data, there is no compelling evidence for either transience, or trap-dependence.

What else can we use the component tests for? As described above, we've used the sum of **TEST3** and **TEST2** test statistics, divided by model df, to derive an estimate of $\hat{c}$. But, remember that the model we're testing here is the fully time-dependent CJS model (i.e., $\{\varphi_t p_t\}$). But, what do we do if the time-dependent CJS model isn't our general model - what if we want to 'start' with some other model? As it turns out, the components of **TEST 3** and **TEST 2** are still useful in assessing fit, and providing estimates of $\hat{c}$, for a variety of models. The following table indicates some of the ways in which components can be used in this way. Note that we haven't covered some of these models yet (but will in later chapters).

components used	*model*	*detail*
TEST3.SR+TEST3.SM+TEST2.CT+TEST2.CL	$\varphi_t p_t$	fully time-dependent CJS model
TEST3.SM+TEST2.CT+TEST2.CL	$\varphi_{a2-t/t} p_t$	two age-class for survival, time-dependence in both age-classes (also know as the 'transience' models in some references)
TEST3.SR+TEST3.SM+TEST2.CL	$\varphi_t p_{t*m}$	immediate trap-dependence in recapture rate (see Pradel 1993)

Thus, using **RELEASE**, and **U-CARE**, goodness-of-fit tests are available readily for 3 models - which, as it turns out, are often the starting points for many single-site recapture analyses. Among them, $\{\varphi_t p_t\}$, which makes the assumptions that survival and capture probabilities are solely time-dependent,

is the most restrictive because it does not permit survival to differ between newly marked and previously marked animals contrary to $\{\varphi_{a2-t/t}p_t\}$, nor capture to differ between animals captured at the previous occasion and those not captured then, contrary to model $\{\varphi_t p_{m*t}\}$. In fact, $\{\varphi_t p_t\}$ is nested within each of the two other models. Models $\{\varphi_t p_{m*t}\}$ and $\{\varphi_{a2-t/t}p_t\}$ are not directly related (i.e., are not nested).

As a consequence of this hierarchy, the goodness-of-fit test of $\{\varphi_t p_t\}$ involves more component tests (because more assumptions are to be checked) than the goodness-of-fit tests of $\{\varphi_t p_{m*t}\}$ or $\{\varphi_{a2-t/t}p_t\}$. In fact, the goodness-of-fit test of $\{\varphi_t p_t\}$ can be decomposed into two steps, in either of two ways:

1. via $\{\varphi_{a2-t/t}p_t\}$: the goodness-of-fit test of $\{\varphi_{a2-t/t}p_t\}$; then, if and only if $\{\varphi_{a2-t/t}p_t\}$ appears valid, the test of $\{\varphi_t p_t\}$ against $\{\varphi_{a2-t/t}p_t\}$
2. via $\{\varphi_t p_{m*t}\}$: the goodness-of-fit test of $\{\varphi_t p_{m*t}\}$; then, if and only if $\{\varphi_t p_{m*t}\}$ appears valid, the test of $\{\varphi_t p_t\}$ against $\{\varphi_t p_{m*t}\}$

Thus, **RELEASE** and (especially) **U-CARE** provide very good capabilities for GOF testing for several important live-encounter 'mark-recapture' models. But, notice that the models being tested are all 'time-dependent'. While it is true that in many cases the most general model in a candidate model set (and the model for which $\hat{c}$ is estimated) is a time-dependent model, this is not always the case. What if your data are too sparse to ever support a time-dependent model? Or, what if your data don't involve live encounter data? Is there a more generic approach to GOF testing that can be used for any kind of data?

At the risk of oversimplifying, we note that GOF testing for 'typical' data from marked individuals is a form of contingency analysis - do the frequencies of individuals exhibiting particular encounter histories match those expected under a given null model, for a given number released on each occasion? You have probably already had considerable experience with some forms of GOF testing, without knowing it. For example, in some introductory class you might have had in population genetics, you might recall that deviations from Hardy-Weinberg expectations were 'established' by GOF testing - through comparison of observed frequencies of individual genotypes with those expected under the null model.

In general, the goodness-of-fit of the global model can be evaluated in a couple of ways: traditionally, by assuming that the deviance for the model is χ^2 distributed and computing a goodness-of-fit test from this statistic, and using **RELEASE** (for live recapture data only) to compute the goodness-of-fit tests provided by that program (as described previously). However, this approach is generally not valid because the assumption of the deviance being χ^2 distributed is seldom met, especially for multinomial data. Program **RELEASE**, which is only applicable to live recapture data, or dead recovery data under some simplifying assumptions, suffers from the same problem to some degree - but usually lacks statistical power to detect lack of fit because of the amount of pooling required to compute χ^2 distributed test statistics. **RELEASE** also is only really appropriate for simple variations of the time-dependent CJS model.

An alternative, and conceptually reasonable approach, is to use an approach based on 'resampling' the data. In addition to providing a basic GOF diagnostic, such approaches also enable you to 'estimate' the magnitude of the lack of fit. As we shall see, this lack of fit becomes important in assessing 'significance' of some model comparisons. In the following sections, we'll introduce two resampling-based approaches to GOF testing and estimation of $\hat{c}$ currently available in **MARK**: the bootstrap, and the median $\hat{c}$.

5.6. MARK and bootstrapped GOF testing

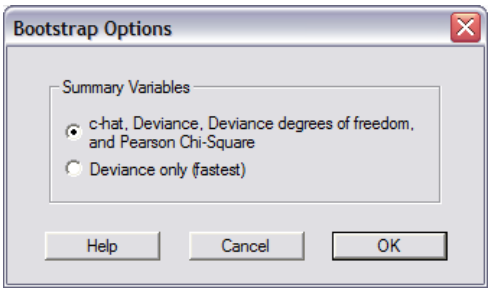
As noted in the **MARK** help file, with the bootstrap procedure, the estimates of the model being evaluated for goodness of fit are used to generate data, i.e., a parametric bootstrap. In other words, the parameter estimates (survival, recapture, recovery rate...) for the model in question are used to simulate data. These simulated data exactly meet the assumptions of the model, i.e., no over-dispersion is included, animals are totally independent, and no violations of model assumptions are included. Data are simulated based on the number of animals released at each occasion. For each release, a simulated encounter history is constructed. As an example, consider a live recapture data set with 3 occasions (2 survival intervals) and an animal first released at time 1. The animal starts off with an encounter history of 1, because it was released on occasion 1. Does the animal survive the interval from the release occasion until the next recapture occasion? The probability of survival is φ_1, provided from the estimates obtained with the original data. A uniform random number in the interval [0, 1] is generated, and compared to the estimate of φ_1. If the random number is less than or equal to φ_1, the animal is considered to have survived the interval. If the random value is greater than φ_1, the animal has died. Thus, the encounter history would be complete, and would be '100'. Suppose instead that the animal survives the first interval. Then, is it recaptured on the second occasion? Again, a new random number is generated, and compared to the capture probability p_2 from the parameter estimates of the model being tested. If the random value is less than p_2, the animal is considered to be captured, and the encounter history would become 110. If not captured, the encounter history would remain 100. Next, whether the animal survives the second survival interval is determined, again by comparing a new random value with φ_2. If the animal dies, the current encounter history is complete, and would be either '100' or '110'. If the animal lives, then a new random value is used to determine if the animal is recaptured on occasion 3 with probability p_3. If recaptured, the third occasion in the encounter history is given a '1'. If not recaptured, the third occasion is left with a zero value.

Once the encounter history is complete, it is saved for input to the numerical estimation procedure. Once encounter histories have been generated for all the animals released, the numerical estimation procedure is run to compute the deviance and its degrees of freedom. In other words, suppose there are a total of 100 individuals in your sample. Suppose you are testing the fit of model $\{\varphi_t p_t\}$. What **MARK** does is, for each of these hundred animals, simulate a capture (encounter) history, using the estimates from model $\{\varphi_t p_t\}$. **MARK** takes these 100 simulated capture histories and 'analyzes them' - fits model $\{\varphi_t p_t\}$ to them, outputting a model deviance, and a measure of the lack of fit, $\hat{c}$ (or, 'c-hat'), to a file. Recall from our early discussion of the AIC (Chapter 4, earlier in this chapter) that $\hat{c}$ is the *quasi-likelihood parameter*. If the model fits perfectly, $\hat{c} = 1$. c is estimated (usually) by dividing the model deviance by the model degrees of freedom. The quasi-likelihood parameter was used to adjust AIC for possible overdispersion in the data (one possible source of lack of fit). Later in this chapter, we will discuss the use of this parameter more fully. The entire process of 'simulate, estimate, output' is repeated for the number of simulations requested. When the requested number of simulations is completed, the user can access the bootstrap simulations results database to evaluate the goodness of fit of the model that was simulated.

Let's look at an example of doing this in **MARK**. We will assess the GOF of the fully time-dependent CJS model $\{\varphi_t p_t\}$ to the male European Dipper data set. If you still have the database file from your earlier analyses of this data set go ahead and open it up in **MARK**. If not, start **MARK**, open up a new project using the male Dipper data, and fit model $\{\varphi_t p_t\}$ to the data.

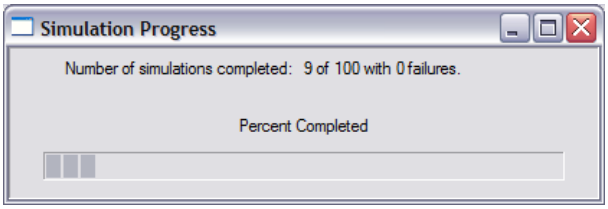
Now, to perform a bootstrapped GOF test on model $\{\varphi_t p_t\}$, highlight it in the results browser by clicking on the model once. Right-click with the mouse, and '**retrieve**' the model. Once you have selected the appropriate model, pull down the '**Tests**' menu, and select '**Bootstrap GOF**'.

You will then be presented with a new window, where you are asked if you want to output just the model deviance from the simulated data, or the deviance, deviance degrees of freedom, and the quasi-likelihood parameter c. As noted in the window, outputting the deviance alone is the fastest, but we suggest that you go for all three - it takes longer computationally, but ultimately gives you all the information you need for GOF testing, as well as a robust estimate of c which, ultimately, is necessary for adjusting the models fits in your analysis.



You will then be prompted for a name for the file into which the bootstrapped estimates are to be output. The default is `BootstrapResults.dbf`.

Finally, you will be asked to specify the number of simulations you want to make (the default is 100), and a random number seed (the default is to use the computer system clock to generate a random number seed). While using the same seed is useful on occasion to diagnose particular problems, in general, you should always use a new random number seed. Once you have entered an appropriate number of simulations (more on this below), and a random number seed, click '**OK**'. **MARK** will then spawn a little 'progress window', showing you what proportion of the requested number of simulations has been completed at a given moment.

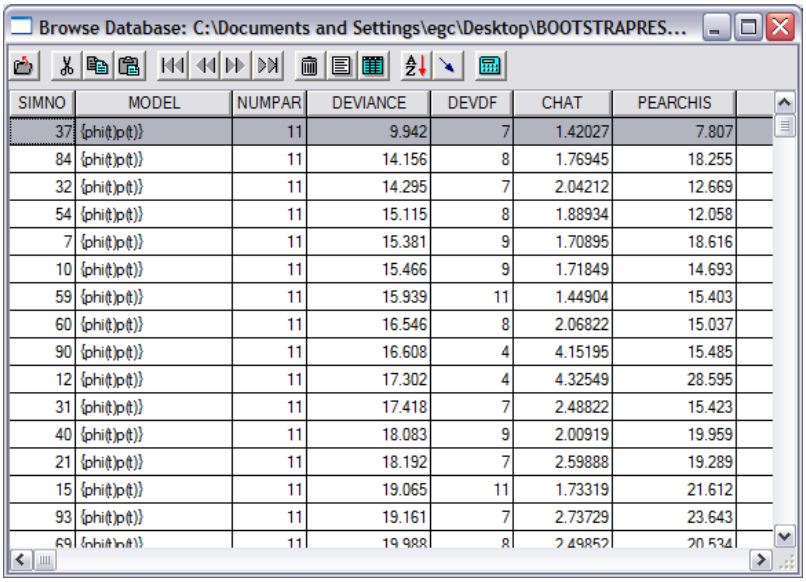


Remember, that for each iteration, **MARK** is (1) simulating capture histories for each individual in the original sample, and (2) fitting model $\{\varphi_t p_t\}$ to these simulated data. As such, it will take some time to complete the task. What do we mean by 'some'? Well, this depends on (1) how big the original data set is, (2) the 'complexity' of the model being fit (i.e., the number of parameters), (3) the number of bootstrapped samples requested, and (4) the computing horsepower you have at your disposal.

OK – now the big question: how many simulations do we need? To answer this question, you first have to understand how we use these simulated data, and the estimated deviances and quasi-likelihood parameters we derived from them. We'll start with the deviances. In essence, what we try to do with the bootstrapped values is to 'see where the observed model deviance falls on the distribution of all the deviances from the simulated data'. In other words, plot out the distribution of the deviances from the data simulated under the model in question, and look to see where the observed deviance - the deviance from the fit of the model to the original data - falls on this distribution. Suppose for example, the deviance of the original model was 104.5, whereas the largest deviance from 1000 simulations was

only 101.2. Then, you would conclude that the possibility of observing a deviance as large as 104.5 was less than 1/1000 (i.e., $P < 0.001$). Or, suppose that you sorted the deviances from the simulated data, from lowest to highest, and found that the 801th deviance was 104.1, and the 802nd value was 105.0. In this case, you would conclude that your observed deviance was 'reasonably likely' to be observed, with a $P < 0.198$ (198/1000), because 198 of the simulated values exceeded the observed value.

MARK makes it easy to do this. Once your simulations are complete, pull down the '**Simulations**' menu, and select '**View Simulation Results**'. You will then be asked to pick the file containing the results of the simulations (the default was '`BootstrapResults.dbf`'). Select the file. A window will pop up that bears a fair resemblance to an Excel spreadsheet (in fact, you can read it into Excel if needed).



SIMNO	MODEL	NUMPAR	DEVIANCE	DEVDF	CHAT	PEARCHIS
37	{phi(t)p(t)}	11	9.942	7	1.42027	7.807
84	{phi(t)p(t)}	11	14.156	8	1.76945	18.255
32	{phi(t)p(t)}	11	14.295	7	2.04212	12.669
54	{phi(t)p(t)}	11	15.115	8	1.88934	12.058
7	{phi(t)p(t)}	11	15.381	9	1.70895	18.616
10	{phi(t)p(t)}	11	15.466	9	1.71849	14.693
59	{phi(t)p(t)}	11	15.939	11	1.44904	15.403
60	{phi(t)p(t)}	11	16.546	8	2.06822	15.037
90	{phi(t)p(t)}	11	16.608	4	4.15195	15.485
12	{phi(t)p(t)}	11	17.302	4	4.32549	28.595
31	{phi(t)p(t)}	11	17.418	7	2.48822	15.423
40	{phi(t)p(t)}	11	18.083	9	2.00919	19.959
21	{phi(t)p(t)}	11	18.192	7	2.59888	19.289
15	{phi(t)p(t)}	11	19.065	11	1.73319	21.612
93	{phi(t)p(t)}	11	19.161	7	2.73729	23.643
69	{phi(t)p(t)}	11	19.988	8	2.49852	20.534

In the spreadsheet, you will see the number of the simulation, the name of the model being simulated, the number of estimable parameters in the model (in this case, the number of parameters is the number determined by the rank of the variance-covariance matrix - see the addendum in Chapter 4 for technical details). Next, the model deviance, and depending upon which option you selected when you ran the simulations, the deviance degrees of freedom and the $\hat{c}$ values. To sort the data in order of ascending deviances, simply click the '**A-Z**' icon on the toolbar, and then select the deviances (you can sort by any or all the elements in the spreadsheet - we're only after the deviance at this stage).

First, the deviances of the simulated data can be ranked (sorted into ascending order), and the relative rank of the deviance from the original data determined. In this example, we note from the results browser that the deviance for model $\{\varphi_t p_t\}$ for the male dipper data is 36.401. Sorting the deviances from the simulated data, we find that the 917th deviance is 36.344, while the 918th deviance is 36.523. The rank of the sorted deviances can be determined using one of the tools on the spreadsheet toolbar.

Thus, the probability of a deviance as large or greater than the observed value of 36.401 is approximately 0.082. So, depending upon your 'comfort level' (after all, selection of an α-level is rather arbitrary), there is probably fair evidence that model $\{\varphi_t p_t\}$ adequately fits the male Dipper data. On the other hand, some people might argue (reasonably) that $P = 0.082$ isn't particularly 'comforting', so perhaps in fact there is some evidence of lack of fit.

However, this leads us back to the following question - how many simulations do you need? In our experience, a two-stage process generally works well. Run 100 simulations, and do a rough comparison

of where the observed deviance falls on the distribution of these 100 values. If the 'P-value' is > 0.2, then doing more simulations is probably a waste of time - the results are unlikely to change much (although obviously the precision of your estimate of the P-value will improve). However as the value gets closer to nominal significance (say, if the observed P-value is < 0.2), then it it probably worth doing $\gg 100$ simulations (say, 500 or 1000). Note that this is likely to take a **long** time (relatively speaking, depending on the speed of your computer).

What else can we do with these bootstrapped simulations? Well, perhaps the most useful thing we can do is to estimate the over-dispersion parameter, c. Why? Well, recall that if the model fits the data 'perfectly', then we expect the value of $\hat{c}$ to be 1.0; $\hat{c}$ is estimated as the ratio of the model χ^2 divided by the model df. When the value of $\hat{c} > 1$, this is consistent with the interpretation that there is some degree of overdispersion. With a $P = 0.082$, perhaps we might believe that there is some marginal evidence for lack of fit of the general model to the data. As noted in the **MARK** helpfile, two approaches are possible, based on the deviance directly, and on $\hat{c}$. For the approach based on deviance, the deviance estimate from the original data is divided by the mean of the simulated deviances to compute $\hat{c}$ for the data. The logic behind this is that the mean of the simulated deviances represents the expected value of the deviance under the null model of no violations of assumptions (i.e., perfect fit of the model to the data). Thus, $\hat{c}$ = observed deviance divided by the expected deviance provides a measure of the amount of over-dispersion in the original data. The second approach is to divide the observed value of $\hat{c}$ from the original data by the mean of the simulated values of $\hat{c}$ from the bootstraps. Again, we use the mean of the simulated values to estimate the expected value of $\hat{c}$ under the assumption of perfect fit of the model to the data. Mean values of both $\hat{c}$ and deviance are easily obtained by simply clicking the 'calculator' icon on the toolbar of the spreadsheet containing the simulated values.

begin sidebar

Careful!

Remember, the simulation results browser allows you to derive a mean $\hat{c}$ simply by clicking a button. However, remember that this mean $\hat{c}$ value is **not** the $\hat{c}$ you need to use. Rather, if you want to use the bootstrapped estimates of $\hat{c}$'s (the 2nd of the two approaches described above), then you take the *observed* model $\hat{c}$ and divide this value by the *mean* $\hat{c}$ from the bootstraps.

end sidebar

As noted in the **MARK** helpfile, there is no good understanding at present of the relative merits of these two approaches. For the example of the male Dipper data, using the observed deviance divided by the mean deviances of the simulated data yields a value of (36.401/25.673) = 1.418. To use the second approach, we first derive the observed $\hat{c}$ value - the model deviance divided by the deviance degrees of freedom. While the model deviance can be read directly from the results browser, the deviance degrees of freedom is obtained by looking in the complete listing of the estimation results - immediately below the print-out of the conditioned S-vector (which is described in the addendum to Chapter 4) In this example, observed $\hat{c}$ is (36.401/7) = 5.20. Dividing this observed value by the mean $\hat{c}$ from the bootstrapped simulations yields (5.20/3.396) = 1.531, which is slightly higher than the value obtained dividing the observed deviance by the mean deviance.

Which one to use? Until more formal work is done, it probably makes sense to be conservative, and use the higher of the two values (better to assume worse fit than better fit - the further $\hat{c}$ is from 1, the bigger the departure from 'perfect fit'). On a practical note, because the observed deviance divided by the mean of the bootstrap deviances does not rely on estimating the number of parameters, it is typically much faster. '**`Bootstrap Options`**' allows you to specify that you are only interested in the deviance, and not $\hat{c}$, from the bootstrap simulations. Generally, the results are often about the same between the two approaches, but can be different when the degrees of freedom of the deviance varies a lot across the

bootstrap simulations (caused by a small number of releases).

5.6.1. RELEASE versus the bootstrap

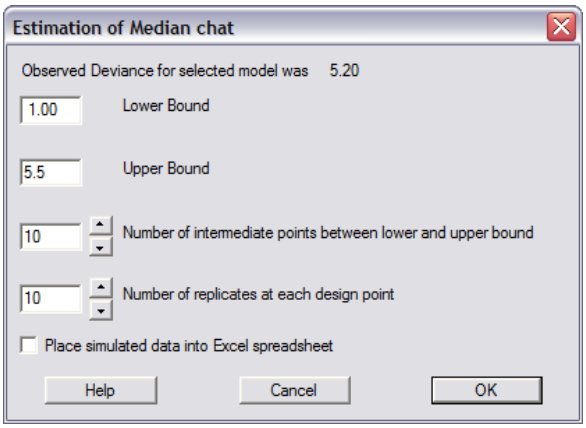
When 'true' $\hat{c}$ is 1 (i.e., no extra-binomial variation), both **RELEASE** and the bootstrap do equally well (equivalent bias, which was very low in both cases). However, when data were simulated with a 'true' $\hat{c}$ of 2 (i.e., considerable extra-binomial variation), the bootstrap was found to perform less well than did **RELEASE** (negatively biased), with the magnitude of the bias increasing with increasing numbers of occasions.*

This seems to imply that **RELEASE** is your best option. Arguably, it might be for standard capture-recapture data (live encounters), but will clearly be of limited utility for data types which are not consistent with **RELEASE** (live encounter/recapture data only). So, are we stuck? Well, perhaps not entirely.

5.7. 'median $\hat{c}$' - a way forward?

A new approach (which has been implemented in **MARK**) has recently been described, and appears quite promising. As with all good ideas, it is based on a simple premise: that the best estimate of $\hat{c}$ is the value for which the observed 'deviance $\hat{c}$' value (i.e., the model deviance divided by the model degrees of freedom) falls exactly half-way in the distribution of all possible 'deviance $\hat{c}$ values generated (simulated) under the hypothesis that a given value of c is the true value. As such, 50% of the generated 'deviance $\hat{c}$' values would be greater than the observed value, and 50% would be less than the observed value. The half-way point of such a distribution is the 'median', and thus, this new approach to GOF testing is referred to in **MARK** as the 'median $\hat{c}$' approach. We'll introduce this approach by means of a familiar example - the male European dipper data set we analyzed in the preceding chapter (`ed_males.inp`). Using program **RELEASE**, the estimate of $\hat{c}$ for our general model $\{\varphi_t p_t\}$ is (9.5598/9) = 1.0622, where 9.5598 is the model χ^2 statistic, and 9 is the model degrees of freedom. Based on a bootstrap GOF test, using 500 bootstrap samples, the estimate of $\hat{c}$ is ~ 1.53 (which is a fair bit larger than the value from **RELEASE**).

Now, what about this new approach - the 'median $\hat{c}$'? Well, to run the median GOF test, we simply select this option from the '**Tests**' menu. Doing so will spawn the following new window:



* For details, see White, G. C. (2002) *Journal of Applied Statistics*, **29**, 103-106.

At the top, the observed deviance is noted: 5.20 (actually, it's the *observed deviance* $\hat{c}$: the model deviance divided by the deviance degrees of freedom: (36.401349/7) = 5.20). Next, you're presented with a lower and upper bound. The lower bound defaults to 1, since a deviance $\hat{c}$ of 1.0 indicates 'perfect' fit of the model to the data. The upper bound (5.5) is slightly larger than the observed deviance $\hat{c}$.

Next, you're asked to specify the number of intermediate 'design' points between the lower and upper bound, and the number of replicates at each 'design point'.

What do these refer to? Well, first, **MARK** is going to (ultimately) fit a regression line to some simulated data - for a series of different values of $\hat{c}$ (i.e., the number of intermediate points), simulate some data - each time you do so, output the calculated deviance $\hat{c}$ for those simulated data. The number of 'replicates' is the number of simulations you do for each value of c you simulate, between the lower and upper bound. Just like with all regressions, the more points you have between the lower and upper bound, and the greater the number of replicates at each point, the better the precision of your regression. **MARK** defaults to 10 for each, since this represents a good compromise in most cases between precision (which is always something you want to improve), and time (increasing the number of intermediate points and/or the number of replicates at each design point, will take a very long time to run for most problems – yet another reason to start agitating for a faster computer).

OK, so far, so good. But what is this 'regression' we've mentioned a couple of times? Basically, it's a *logistic regression* – a regression where the response variable is a binary state variable, suitably transformed (usually using the logit transform - hence the name logistic regression). In this case, the binary state is 'above the observed deviance $\hat{c}$' or 'below the observed deviance $\hat{c}$'. So, for each value of $\hat{c}$ in the simulation, we generate a sample of deviance $\hat{c}$'s. We count how many of these values are 'above' or 'below' the observed value, and regress this proportion on $\hat{c}$ (using the logistic regression). Then, all we need to do is use the regression equation to figure out what value of $\hat{c}$ corresponds to the situation where the proportions of the simulated deviance $\hat{c}$'s are equal (i.e., where the number 'above' the observed value is exactly the same as the number 'below' the observed value. This, of course, is the *median* of the distribution). If the number 'above' and 'below' is the same, then this is our best estimate of $\hat{c}$, since values above or below the observed value are equally likely.

begin sidebar

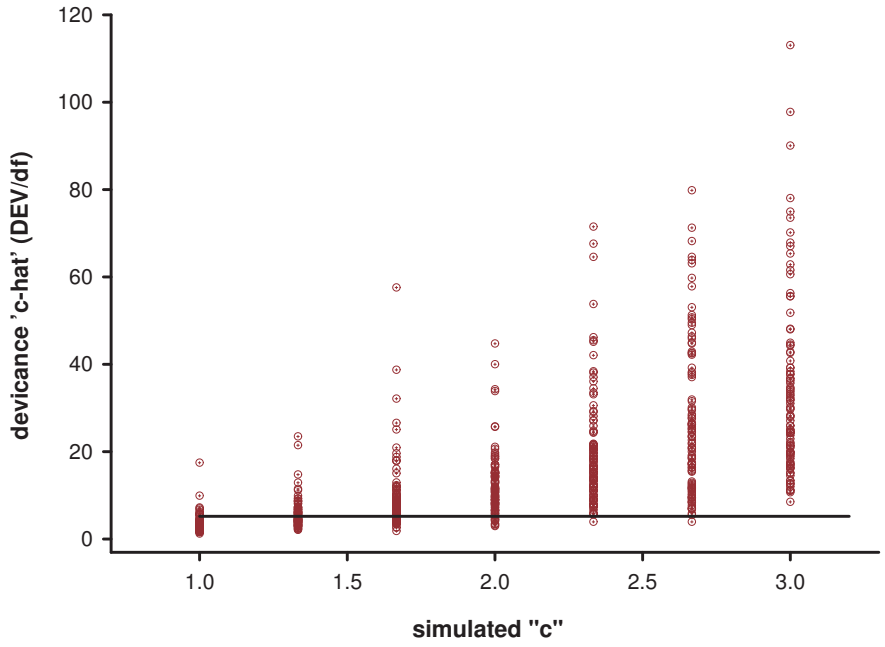
median $\hat{c}$ and logistic regressions in MARK

The logistic regression analysis is performed by **MARK** as a *known-fate model*; known-fate models are discussed in a later chapter. For now, it is sufficient to consider that 'fate' in this case is the 'known' proportion of c values above - or below (doesn't matter) - the observed deviance $\hat{c}$.

Output consists of the estimated value of c and a SE derived from the logistic regression analysis, with these estimates provided in a notepad or editor window preceding the known fate output. In addition, a graph of the observed proportions along with the predicted proportions based on the logistic regression model is provided. The initial dialog box where the simulation parameters are specified also has a check box to request an Excel spreadsheet to contain the simulated values. This spreadsheet is useful for additional analyses, if desired.

end sidebar

Let's try this for the male dipper data. The default upper bound is 5.5. We'll change this upper bound to 3.0, for both convenience (i.e., to save time) and for heuristic reasons (if $\hat{c}$ is > 3.0, we may have more fundamental problems; Lebreton *et al*. 1992). We set the number of intermediate points (i.e., c values) to 3 (so, 5 total design points on the function), and the number of iterations (replicates) at each value of c to 100. If you're trying this on your own, this might take some time. Here is a plot of the $\hat{c}$ values simulated by **MARK** for the male dipper data.



Each of the open circles in the plot represents the deviance $\hat{c}$ for one of the bootstrapped replicates, at each of the 5 design points (i.e., values of c) in the simulation (in this case, 1.0, 1.5, . . . , 3.0). The solid horizontal line represents the observed deviance $\hat{c}$ for the general model (5.2002). Remember, what we're after is the value of c for which the number of open circles above the observed value is equal to the number of open circles below the observed value. From the figure, we see clearly that this occurs somewhere in the range $1.0 \rightarrow 1.5$; for all values of $c > 1.5$, there are clearly far more values above the observed value, than below it.

In fact, if we calculate the frequency of 'above' and 'below' for each value of c, we get the following contingency table (based on 100 replicates in each column – note, your numbers may differ):

	1.0	1.5	2.0	2.5	3.0
above observed	11	66	94	99	100
below observed	89	34	6	1	0

Again, we see clearly that the 'median' will occur somewhere between $\hat{c} = 1.0$ and $\hat{c} = 1.5$. Applying a logistic regression to these data (where the logit transformed proportion is the response variable, and the value of c is the independent variable), we get an estimate of the intercept of $\hat{\beta}_0 = 6.7557$, and an estimate of the slope of $\hat{\beta}_1 = -4.8476$. Since we're interested in the point at which the proportion above and below is equal at 50% (i.e., 0.5), then we simply take the logistic equation, set it equal to 0.5:

$$0.5 = \frac{e^{-4.8476(c)+6.7557}}{1 + e^{-4.8476(c)+6.7557}}$$

Solving for c (not to worry - **MARK** handles all this for you) yields our estimate of $\hat{c} = 1.3936$ (with a SE of 0.033, based on the number of intermediate points and replicates used in our simulation). That's it! So, based on the median approach, the estimate of $\hat{c}$ is 1.3936, which is intermediate between the value reported by **RELEASE**, and the bootstrapped estimate.

So, which one to use? Well, the median $\hat{c}$ GOF approach is a 'work in progress', but preliminary assessment indicates that it appears to work well. In comparisons for the CJS data type to the **RELEASE** model, the median $\hat{c}$ is biased high, as much as 15% in one case of $\varphi = 0.5$ with 5 occasions. However, the median-$\hat{c}$ has a much smaller standard deviation for the sampling distribution than the $\hat{c}$ estimated by **RELEASE**. That is, the mean squared error (MSE) for the median $\hat{c}$ is generally about $\frac{1}{2}$ of the MSE for the **RELEASE** estimator. Thus, on average, the median $\hat{c}$ is closer to 'truth' than the **RELEASE** $\hat{c}$, even though the median $\hat{c}$ is biased high.

One of the current limitations of the median $\hat{c}$ goodness-of-fit procedure is that individual covariates are not allowed - but unlike the bootstrap, it can be used for most of the other models in **MARK**. Further, it can be applied to any model you want - **RELEASE** (and **U-CARE**) requires you to use the fully time-specific model as the general model, which may not always be ideal depending on your particular circumstances.

Overall, the median $\hat{c}$ GOF approach seems promising. However, be advised that no GOF test is perfect - the median GOF approach is a 'work in progress', and its performance compared to other GOF approaches has not been fully evaluated in all situations (e.g., multi-state models). For the moment, we'd suggest, where possible, running as many of the GOF tests as are available – each has different strengths and weaknesses. Hopefully, there will be reasonable agreement among them. If not, then you need to decide on the choice of which $\hat{c}$ estimate to use (largest, smallest, or otherwise, recognizing that the larger the value of $\hat{c}$ used, the more support simpler models will get relative to more complicated models in your candidate model set, and that the relative scaling of AIC itself will change), and the reason for the choice should be communicated.

begin sidebar

The bootstrap and/or median-$\hat{c}$ won't give me an estimate for $\hat{c}$! Now what?

In some cases, when you run either the bootstrap or median-$\hat{c}$ goodness of fit tests, some/many/all of the simulations will have 'problems' - failure to converge, nonsense values, incorrect parameter counts, and so forth. In virtually all cases, this is not a problem with **MARK**, but, rather, with the general model you are trying to fit your data to (and which **MARK** is attempting to use to simulate data).

Remember, that the general approach to model selection involves multi-model inference over the set of candidate models, under the assumption that at least one of the models adequately fits the data. And of course, this is what you are using the GOF test to assess - the adequacy of fit of your 'general model' to the data. In many cases, your general model will be a time-dependent model in one or more of the parameters. However, you need to be aware that for some sparse data sets, a fully time-dependent model is unlikely to fit your data well - many/most of the parameters will be poorly estimated, if they are estimated at all. And, if one or more parameters can't be estimated - because of sparseness in the data - then **MARK** will not be able to successfully simulate data under that model.

The solution is to accept - grudgingly, perhaps - that your data may simply be inadequate to fitting a time-specific general model. There is nothing particularly wrong with that - you simply need to find a more reduced parameter model which satisfies your needs (i.e., which adequately fits the data). Of course, using anything other than a time-dependent model precludes use of **RELEASE** or **U-CARE** for GOF testing, but that is not a particular problem if the goal is estimation of $\hat{c}$ (and not looking in detail at the underlying sources of lack of fit).

end sidebar

Now it is **very important** that you realize that both the bootstrap and the median $\hat{c}$ approaches we've just described assume that the source of lack of fit is simple extra-binomial noise. In fact, this is precisely how the simulations work. For example, to simulate a data set with a $\hat{c} = 2.0$, the frequency of each observed encounter history is simply doubled.

What this means is that the estimated $\hat{c}$ is robust (more or less) if and only if the primary source of lack of fit is extra-binomial. If the lack of fit is due to something else (say, structural problems in the model), then the estimated $\hat{c}$ may not be particularly useful. Some of these issues are addressed in the following section.

5.8. What to do when the general model 'doesn't fit'?

We began this chapter by noting that model selection (whether it be by AIC, or some other procedure) is conditional on adequate fit of a general model to the data. As such, the ability to assess goodness of fit is important. As mentioned earlier, there are at least 2 classes of reasons why a model might not adequately fit the data. The first is the 'biologically interesting' one - specifically, that the structure of the general model is simply inappropriate for the data. In subsequent chapters, you'll see how we might have to radically alter the basic ultrastructure of the model to accommodate 'biological reality'. For example, if you are marking both juveniles and adults, then there is perhaps a reasonable expectation that their relative survival rates may differ, and thus, one of the assumptions of CJS (specifically assumption 2) - that every marked animal in the population immediately after time (i) has the same probability of surviving to time (i+1) has been violated. The second, perhaps less interesting reason is the possibility that the data are over or under-dispersed for the CJS model – extra-binomial variation. In this section, we will briefly discuss both cases.

5.8.1. Inappropriate model

Your most immediate clue to lack of fit will be a high $\hat{c}$ value. The 'challenge' will be to determine whether or not this is because you have an inappropriate model, or extra-binomial 'noise'. In some cases, this is fairly straightforward. For example, to confirm that the basic live-encounter CJS model has been rejected because the model is 'biologically unrealistic' for your data, you simply need to carefully examine the detailed **TEST 2** and **TEST 3** contingency tables in your **RELEASE** output file (or, more conveniently, look at it directly with **U-CARE**). What are you looking for? Well, in general, the thing you're looking for is a 'systematic' rejection (or bias) in the individual tables. You need to see if the failure of **TEST 2** or **TEST 3** is 'driven' by a few 'strange' batches, or is due to a 'systematic' bias. What do we mean by 'systematic' bias? Well, by 'systematic', we refer to a bias which occurs consistently at each occasion - a bias in the sense that a particular cell (or cells) in one of the test tables is consistently over or underpopulated.

An example will help make this clear. Suppose you run **RELEASE**, and find that **TEST 3** is rejected, but not **TEST 2**. You say to yourself, 'OK, recapture seems to be OK, but something is wrong with the survival assumption, under the CJS model'. You proceed to look carefully at each of the **TEST3** tables for each batch. You note that **TEST3.SR** is rejected, but that **TEST3.SM** is accepted. Now, what does this mean? Recall that **TEST3.SR** simply asks, of those individuals seen either on or before occasion (i), what proportion were ever seen again? If **TEST3.SR** is rejected, then this suggests that there is a difference in 'survival' among individuals, depending on whether or not they were seen for the first time either on or before occasion (i). However, **TEST3.Sm** only looks at individuals who WERE seen again. Among these individuals, when they were seen again does not depend on whether or not they were seen for the first time at occasion (i).

Suppose you look at each individual **TEST3.SR** table, and find the following - a '+' indicates more individuals than expected (based on the null hypothesis of no differences between groups), and a '-' indicates fewer individuals than expected (under the same hypothesis). Since **U-CARE** provides you with the overall 'directional' test, you can make this determination very quickly. Let's say we have 10 occasions, and we find that this pattern seems to be present in the majority of them (you might use some statistical test, for example a sign test, to determine if the frequency of tables exhibiting a particular pattern occurs more often than expected by random chance). Say, 8/10 contingency tables show this pattern (which will clearly be statistically significant). What does this suggest? Well, among individuals seen for the first time at occasion (i), significantly more are never seen again than expected, relative to individuals who had been seen before occasion (i). In other words, newly marked individuals showed a consistently lower probability of ever being seen again than previously marked individuals.

What could lead to this pattern? One possibility we suggested at the beginning of this section was age effects. Lower survival of newly marked juveniles (relative to adult survival) would lead to this pattern in **TEST3.SR**. Is this the 'only' plausible explanation? Unfortunately, no. Life would be simpler if there was only ever one possible explanation for anything, but, this is generally not the case. This example is no exception. Rejection of **TEST3.SR** could also reflect (1) a marking effect (where the act of marking causes an increase in immediate mortality), (2) presence of transients (migratory individuals leaving the sampling area shortly after marking), or (3) heterogeneity in capture rates (some individuals have low capture rates, some high).

The point here is that there may be more than one possible answer - it is at this point you'll need to use your 'biological insight' to help differentiate among the possible explanations. The other, perhaps more important point is that the presence of a consistent difference in one of the major tests (**TEST2.Ct**, **TEST2.Cm**, **TEST3.SR**, and **TEST3.Sm**) each suggest the possibility of one or more effects which violate the basic CJS assumptions. You will have to rely on your insight to help you identify possible 'biological' explanations for violation of any of these 4 tests - each of them might refer to something completely different.

What can you do if you do reject CJS? Well, the solution depends on what, exactly, 'has gone wrong'. In general, if the individual **TEST 2** or **TEST 3** results seem to show systematic deviations among occasions, the most likely solution will be to reject the CJS model as the 'correct' starting model for your analysis - it clearly doesn't fit, because the inherent assumptions aren't met by the data. In this case, where **TEST3.SR** is rejected, but the other tests are accepted, then the solution is to add age-structure to the model (this will be presented in Chapter 8).

However, simply recognizing that a 'different' starting model (say, a 2-age class model) may be more appropriate is only the first step. You still need to confirm that the data fit your 'new' model. You must go through analogous GOF tests for the 'new' starting model, just as we have done for the CJS model (as discussed earlier in this chapter).

What if your data type is one that can't be handled by **RELEASE** (which, in effect, is every data type other than live-encounter mark-recapture data)? One option is to examine *deviance residual plots*. If you click the '**`Graph Deviance Residuals`**' button on the main **MARK** toolbar, residuals are plotted against their capture history number to assess the fit of the model. The default for the residual plot icon is deviance residuals. However, either deviance residuals or Pearson residuals are available from the '**`Output | Specific Model Output`**' menu of the Results Browser.

A *deviance residual* is defined as

$$\text{sign (obs-exp)} \sqrt{\frac{2\,[(\text{exp-obs}) + \text{obs} \times \ln(\text{obs/exp})]}{\hat{c}}}$$

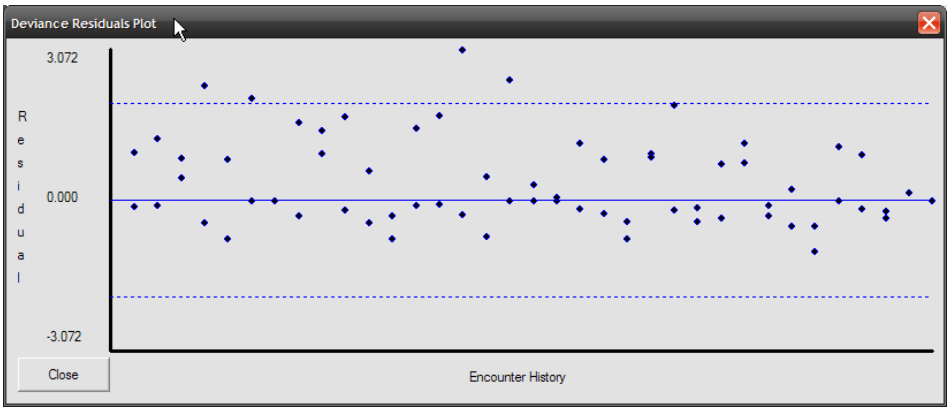
where 'sign' is the sign (plus or minus) of the value of (obs-exp).

A *Pearson residual* is defined as

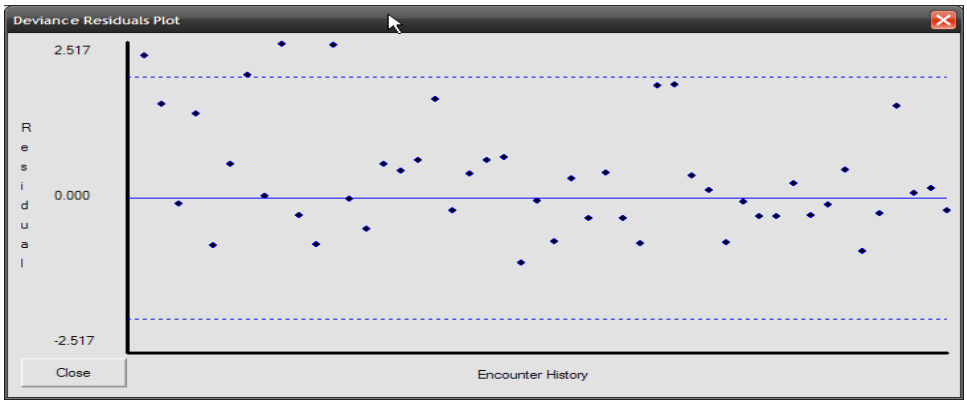
$$\frac{(\text{obs-exp})}{\sqrt{(\text{exp} \times \hat{c})}}$$

Take for example the swift analysis we introduced in Chapter 4. If we run a bootstrap analysis on our general model $\{\varphi_{c*t} p_{c*t}\}$, we get an estimate of $\hat{c} = 1.00$. So, pretty good fit!

This is reflected in the deviance residual plot shown at the bottom of this page. If the model was a 'good-fitting model' (as suggested by our estimated $\hat{c}$), we would expect that there would be no 'trend' (non-randomness) in the pattern of the residuals - roughly half of the residuals should be above the 0.000 line, and half should be below. Further, if there is no extra-binomial variation, then most of these residuals should be fairly close to the 0.0000 line (between the two horizontal dashed lines in the preceding figure). In the preceding plot, there is little apparent trend, although most of the extreme residual are 'on the high side' (large positive deviance - to see which observation is causing a particular residual value, you can place your mouse cursor on the plotted point for the residual, and click the left button. A description of this residual will be presented, including the group that the observation belongs to, the encounter history, the observed value, the expected value, and the residual value). However, the residuals are roughly randomly distributed above and below 0.



Here is an example of a deviance residual plot which seems to indicate lack of fit.



In the second plot, notice both a clear asymmetry in the residuals (greater proportion of positive to negative residuals) as well as some suggestion of a trend (although most residuals are positive, the magnitude of the deviation above zero seems to decline from left to right). Both suggest strongly that there is a structural problem with the fitted model. In fact, this particular plot was generated by fitting a $\{\varphi_{\cdot}\}$ model structure to data where in fact φ increased linearly over time (a topic we discuss in Chapter 6). So, clearly, the fit model was not structurally appropriate, given the underlying model used to generate the data. We will re-visit the use of residual deviance plots to help evaluate lack of fit as we introduce new data types.

5.8.2. Extra-binomial variation

If there are systematic deviations in your contingency tables, or if there is some other indicator of structural causes of lack of fit (say, from a deviance residual plot), then changing the structure of the general model is the appropriate step to take next. However, what do you do, if the deviations in the contingency tables are not 'consistent' among batches - what if (say) 5/10 tables are biased in one direction, and 5/10 are biased in the other direction?

In such cases, where there seems to be no clear 'explanation' (biological or otherwise) for the violation of **TEST 2** or **TEST 3**, you then have only a few options. As you've no doubt gathered if you've read this far into this chapter, the most common 'solution' (although it is only a partial solution) is to 'adjust' your statistics to account for the 'extra noise', or (for the statistically inclined) the extra-binomial variation. Remember that the conceptual basis of all models is 'data = structure + residual variation'. In general, the structure of the residual variation is unknown, but for multinomial distributions, it is known. If the model structure is 'correct', then the variance of the residual 'noise' is 1 (where variance is defined as the expected value of the GOF χ^2 divided by its degrees of freedom). However, even if the residual variation > 1, the structural part of the model can be 'correct'. In simplest terms, if there is 'excess variation' it will show up in the model GOF χ^2 (since this value is essentially a 'residual SS'). Thus, what we need to do is 'correct' everything for the magnitude of this extra variation. To do this, we derive what is known as a variance inflation factor, $\hat{c}$. The larger the value of $\hat{c}$, the greater the amount of 'extra' variation. We have already presented this basic idea earlier in the chapter, as well as the mechanics of how to get **MARK** to adjust things.

Consider, for example, the dipper data set, where we can estimate $\hat{c}$ in several different ways: using the bootstrap, median $\hat{c}$, or by using a χ^2/df approach in **RELEASE** and **U-CARE**. Recall that from our bootstrap fit of $\{\varphi_t p_t\}$ to the male European Dipper data set yielded an estimate of $\hat{c}$ of either 1.418, or 1.531 (depending on whether or not you calculated $\hat{c}$ using the bootstrapped distribution of deviances, or $\hat{c}$ values). The mean of these two values is 1.475. Now, in both **RELEASE** and **U-CARE**, the sum of the overall results of **TEST 2** and **TEST 3** is (in effect) the overall model χ^2. This is provided for you directly in the **RELEASE** and **U-CARE** output. For the male dipper data, **RELEASE** gives the sum of **TEST 2** + **TEST 3** = 9.5598. The model df = 9, and thus from **RELEASE**, $\hat{c}$ is estimated as (**TEST 2** + **TEST 3**)/df = (χ^2/df) = 1.0622. From **U-CARE**, (**TEST 2 + TEST 3**)/df = (χ^2/df) = (11.0621/9) = 1.229. Now, in fact, there does seem to be some variation in estimates of $\hat{c}$, depending on the method selected, with the estimates from the bootstrap approach being the highest, and that from program **RELEASE** being the lowest.

In fact, the reason (in this example) is because the male dipper data has 'a problem' - the data are somewhat sparse - so much so that many of **RELEASE** tests (especially **TEST 3**) are reported as 'invalid' (insufficient data to make the particular contingency test(s) valid). This is also reflected in the fact that one parameter (recapture rate p_3) that is not particularly well-estimated (this can either be because the parameter is estimated near the boundary, or because of 'weirdness' in the data). However, these

nuances notwithstanding, if you have good data (i.e., not so sparse that **RELEASE** has to do a lot of pooling over cells in the contingency tables), then the estimate of $\hat{c}$ using the bootstrap or the median-$\hat{c}$ in **MARK** should be very close to the estimate using the (χ^2/df) approach in **RELEASE** or **U-CARE**.

OK - so now what do we do with our measure of the fit of the CJS model to the male Dipper data? Our estimate of the significance of departure from 'adequate fit of the model to the data' was $P = 0.08$. As noted, our estimate of $\hat{c}$ varied depending upon how it was derived: for now, we'll use $\hat{c}$=1.531 (the value from the bootstrap). Now what? Well, think a moment about what we're doing when we do the model fitting - we want to compare the *relative* fit of different models in the model set. If there is 'significant lack of fit', then intuitively this may influence our ability to select amongst the different models. Also intuitively, we'd like to adjust our model fits to compensate for the lack of fit. How do we accomplish this?

First, look at the 'original results':

Results Browser: Live Recaptures (CJS)

Model	AICc	Delta AICc	AICc Weight	Model Likelihood	No. Par.	Deviance
{phi(.)p(.)}	322.5527	0.0000	0.96003	1.0000	2	41.8147
{phi(t)p(.)}	330.0567	7.5040	0.02253	0.0235	7	38.8147
{phi(.)p(t)}	330.6794	8.1267	0.01650	0.0172	7	39.4374
{phi(t)p(t)}	336.4343	13.8816	0.00093	0.0010	11	36.4013

These AIC_c weights were calculated using the default $\hat{c}$ value of 1.0. As such, we would have concluded that the best model in the model set was ~ 43 times better supported by the data than was the next best model.

What happens if we adjust the values in the results browser using $\hat{c}$=1.51? **MARK** makes such an 'adjustment' very easy to do. Simply pull down the '**`Adjustment`**' menu, and select '**`c-hat`**'. Enter the new value for $\hat{c}$ (in this example, use 1.51). As soon as you've entered the new $\hat{c}$, the various AIC and AIC weighting values in the results browser are converted to $QAIC_c$ values (note that the column labels change in the results browser to reflect this change).

For example, consider the original AIC and AIC weight values for the 4 simplest models we applied to these data:

Results Browser: Live Recaptures (CJS) c-hat = 1.5100000

Model	QAICc	Delta QAICc	QAICc Weight	Model Likelihood	No. Par.	QDeviance
{phi(.)p(.)}	214.9819	0.0000	0.97462	1.0000	2	27.6919
{phi(t)p(.)}	223.4992	8.5173	0.01378	0.0141	7	25.7051
{phi(.)p(t)}	223.9115	8.9296	0.01122	0.0115	7	26.1175
{phi(t)p(t)}	230.6919	15.7100	0.00038	0.0004	11	24.1069

Now, we see that the degree of support for the best model has increased substantially - the best model is now > 70 times better supported than the next best model. In this case, it didn't change our final conclusions, but it is not hard to imagine how changes of this magnitude could make a significant

difference in the analysis of the data.

Does anything else happen to 'our results'? The answer is 'yes', but it isn't immediately obvious - adjusting $\hat{c}$ also changes the estimated standard errors for each of the parameters in each of the models. Try it and confirm this for yourself. However, there is a subtle catch here - the estimated standard errors are changed **only** in the output of the estimates, **not** in the general output. Don't ask! :-)

5.9. How big a $\hat{c}$ is 'too big?'

When should you apply a new $\hat{c}$? Is 1.51 really different than the null, default value of 1.0? At what point is $\hat{c}$ too large to be useful? If the model fits perfectly, then $\hat{c} = 1$. What about if $\hat{c} = 2$, or $\hat{c} = 10$? Is there a point of diminishing utility? As a working 'rule of thumb', provided $\hat{c} \leq 3$, you should feel relatively safe (see Lebreton *et al*. 1992 - pp. 84-85). However, there are a couple of fairly important 'philosophical' considerations you'll need to keep in mind. First, for some analysts, the most important thing is adjusting for fit to make the parameter estimates as robust and valid as possible. However, for others, the question of 'why' there is lack of fit is important. Consider, for example, the standard 'CJS assumptions'. In particular, the assumption that all individuals are equally likely to be caught. In truth, there may very often be considerable variation among individuals in the probability of capture - a clear violation of the CJS assumptions. The question, then, is whether the lack of fit is due to these sorts of violations (strictly speaking, this is referred to as the problem of individual heterogeneity in capture probability), or structural problems with the general model. Clearly, by adjusting $\hat{c}$, you can accommodate lack of fit up to a certain degree, but if the $\hat{c}$ is fairly large (> 2) then this might be a good indicator suggesting a careful examination of the structure of the model in the first place is needed. So, yes, you can 'adjust' for lack of fit simply by adjusting the $\hat{c}$ value used. However, be advised that doing so 'blindly' may obscure important insights concerning your data. It is always worth thinking carefully about whether your general model is appropriate. Moreover, as $\hat{c}$ increases $\gg 1$, interpretation of some of the metrics we'll use for model selection (something we cover in later chapters) becomes more complicated.

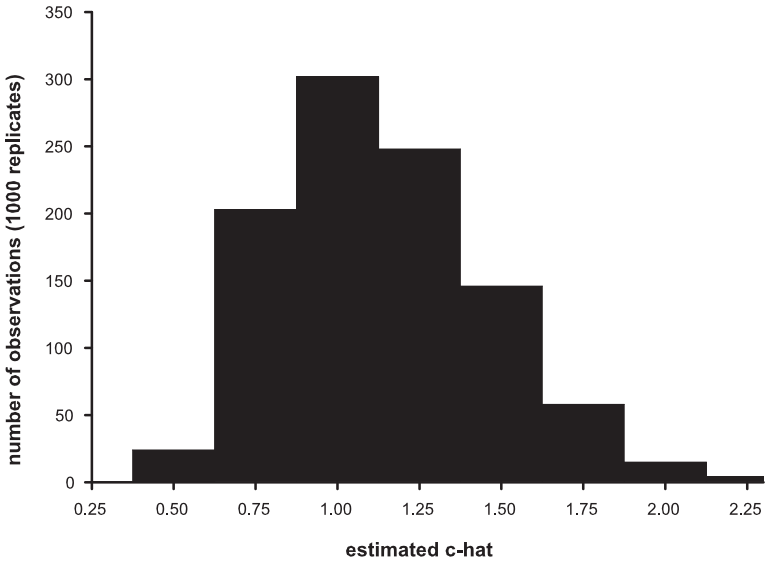
Second, as noted earlier, the bootstrap and median $\hat{c}$ resampling approaches estimate $\hat{c}$ under the assumption that the source of the lack of fit is strictly extra-binomial 'noise' - as you might see, for example, if your sample consisted of male and female pairs, where the fate of one member of the pair was not independent of the other. The more 'behavioral structure' you have in the biology (or process) underlying your data, the more likely this is to be the case. A method to evaluate whether theoretical variance estimates are valid when the individuals are considered statistically independent, or whether variance inflation procedures are required to account for dependence among (say) siblings, is described in Appendix G. However, if the source of lack of fit is structural, and not due to extra-binomial noise, then adjusting for a $\hat{c}$ *estimated* using either of the resampling approaches may not do you much good. In such cases, your best option is to (i) work hard at trying to identify the structural problems in your model, and (ii) see how sensitive your model weights are to manual adjustments (in small increasing increments) in $\hat{c}$. This basic process is also discussed below.

Finally, and perhaps most importantly, remember that we are *estimating* c. And as with any estimate, there is uncertainty about our estimate of $\hat{c}$. As such, if you derive an estimate of c that is (say) $\hat{c} = 1.4$, there may be a significant probability that in fact the true value of c is 1.0 - but, because of the uncertainty in your estimate of c, you can't be sure.

We can demonstrate this fairly easily, using some data simulated under a true model $\{\varphi_t p_t\}$. We will do this using the **RELEASE** simulations capability in **MARK** (see Appendix A). We will simulate 8 occasions in the simulated data set, with 250 newly marked and released individuals on each occasion (clearly, a much bigger data set than the male dipper data). For 1,000 simulations, the mean $\hat{c}$ (estimated

as the **TEST 2** + **TEST 3** χ^2 divided by the df) was 0.997, which is very close to the expected $\hat{c}$ of 1.00. However, a couple of points to make. While the mean is very close to the expected value, there is a fair bit of variation around this value. For example, when we ran the simulation (as described), we get a variance of 0.102, with a 95% CI of [0.543 - 1.57].

This is reflected in the following histogram of $\hat{c}$ estimates:



Note that there is a fair chance that for a given simulated data set, that the observed estimate of $\hat{c}$ is considerably less than, or greater than, 1.0 - even though the true value is 1.0 for these data! This is not really a problem, but one you need to be aware of: it is possible that you are fitting the correct model to the data (such that the true $\hat{c}$ is 1.0), but because each data set you collect is simply one realization of an underlying stochastic probabilistic process, there is some chance that the $\hat{c}$ you observe will differ - sometimes markedly so - from 1. We still use a quasi-likelihood adjustment to account for lack of fit, since we cannot be certain we have the correct model. (*Note*: the histogram should be approximately χ^2 distributed for the given df, as it appears to be for this example).

5.9.1. General recommendations

OK, some general recommendations:

1. identify a general model that, when you fit it to your data, has few or no estimability problems (i.e., all of the parameters seem adequately estimated). Note that this general model may not be a fully time-dependent model, especially if your data set is rather sparse. If there is 'biological' justification for suspecting non-independence among individuals in your data, then you should evaluate whether theoretical variance estimates are valid when the individuals are considered statistically independent, or whether variance inflation procedures are required to account for dependence among (say) siblings. A procedure for performing such an evaluation is introduced in Appendix G.

2. estimate $\hat{c}$ for this general model, using whichever method is most robust for that particular model. While the median $\hat{c}$ approach shows considerable promise to be a generally robust approach to estimate $\hat{c}$, for the moment it is perhaps worth generating $\hat{c}$ using all of the available approaches, and comparing them. If you have some estimates marginally above 1.0, and some marginally below 1.0, it is probably reasonable to assume the $\hat{c} \cong 1.0$. Alternatively, you could choose to be conservative, and select the largest $\hat{c}$, which provides some protection (in a sense) against selecting overly parameterized models.
3. remember that the estimate of c is just that - an *estimate*. Even if the true value of c is 1.0, we use the estimate of $\hat{c}$ to account for model selection uncertainty (since we cannot be certain we have the correct model). Moreover, as noted earlier, the bootstrap and median $\hat{c}$ resampling approaches estimate $\hat{c}$ under the assumption that the source of the lack of fit is strictly extra-binomial 'noise' - where the fate of one individual may not be independent of the other. However, if the source of lack of fit is structural, and not due to extra-binomial noise, then adjusting for a $\hat{c}$ *estimated* using either of the resampling approaches may not do you much good. In such cases, your best option is to (i) work hard at trying to identify the structural problems in your model, and (ii) see how sensitive your model weights are to manual adjustments (in small increasing increments) in $\hat{c}$ (see next point).
4. it is also worth looking qualitatively at the 'sensitivity' of your model rankings to changes in $\hat{c}$. Manually increase $\hat{c}$ in the results browsers from 1.0, 1.25, 1.5 and so on (up to, say, 2.0), and look to see how much the 'results' (i.e., relative support among the models in your candidate model set) changes. In many cases, your best model(s) will continue to be among those with high AIC weight, even as you increase $\hat{c}$. This gives you some grounds for confidence (not much, perhaps, but some). Always remember, though, that in general, the bigger the $\hat{c}$, the more 'conservative' your model selection will be - AIC will tend to favor reduced parameter models with increasing $\hat{c}$ (a look at equation for calculating AIC will show why). This should make intuitive sense as well - if you have 'noise' (i.e., lack of fit), perhaps the best you can do is fit a simple model. In cases where the model rankings change dramatically with even small changes in $\hat{c}$, this might suggest that your data are too sparse for robust estimation, and as such, there will be real limits to the inferences you can make from your candidate model set.

This final point is related to what we might call the 'wince' statement: if you are sure your model structure is correct, and despite you're finer efforts, your $\hat{c}$ is $\gg 3$, or if your model rankings change radically with even small changes in $\hat{c}$, then you might just have to accept you don't have adequate data to do the analysis at all (or, perhaps in a more positive tone, that there are real limits to the inferences you can draw from your data). Unfortunately, your 'time, effort, and expense' are not reasonable justifications for pushing forward with an analysis if the data aren't up to it. Remember the basic credo. . .'garbage in...garbage out'.

Last words - for now (remember, GOF testing is very much a work in progress). If your data are based solely on live encounter data, then it appears that for the moment, for most data sets, using estimates of $\hat{c}$ derived from (χ^2/df) calculations (using either **RELEASE** or **U-CARE**) is more robust (less biased) than bootstrapped or median $\hat{c}$ estimates. But, what if you don't have live encounter data, or perhaps some mixture of live encounter with other types of encounter data (e.g., perhaps dead recovery)? For other data types (or mixtures), in some cases GOF tests have been suggested (e.g., contingency GOF testing for dead recovery data is implemented in program **MULT**), but the degree to which estimated

of $\hat{c}$ from these approaches may be biased is unknown. However, in many cases, the bootstrap or median $\hat{c}$ can be run, which does provide some estimate of $\hat{c}$, albeit potentially biased in some cases.

5.10. Summary

That's the end of our **very** quick stroll through the problem of GOF. In this chapter, we have focussed on using program **MARK** and programs **RELEASE** and **U-CARE** to handle this task. Remember - the bootstrap and median-$\hat{c}$ approaches provides an omnibus technique for assessing GOF for **any** model, as well as providing robust estimates of $\hat{c}$ (at least, in theory – GOF testing is still very much a work in progress). **RELEASE**, which should be applied to live-recapture models only, is a good tool for examining 'where' the departures occur. Residual plots can also be quite helpful.

CHAPTER 6

Adding constraints: MARK and linear models

Up until now, we've used **MARK** to build relatively simple models. We've seen how to use **MARK** to help with model selection, and how the process of model selection can be viewed in an analysis of variance context, by comparing models with and without grouping factors (Chapter 4).

However, suppose, for example, you want to build a model where annual variation in survival is 'related' to some weather variable. How would you construct a model where survival was constrained to be a function of 'weather'? The concept of 'constraints', and how to use them to apply linear models to **MARK**, is one of the most important and powerful extensions of what we have covered so far.

What do we mean by 'constraint'? In the present context, we are referring to a mathematical constraint – 'forcing' **MARK** to estimate survival and recapture probabilities after imposing a specific set of linear constraints on the structure of the underlying model. While the concept is simple, the ability of **MARK** to allow you to construct linear models gives you considerable flexibility in addressing a very large number of questions and hypotheses with your data; if you can conceive of a linear model (ANOVA, ANCOVA, multiple regression etc), you can apply it to mark-encounter data using **MARK**. The only thing you'll need to know is how to construct the 'linear constraint' – the linear model. This is the subject of the present chapter.

6.1. A (brief) review of linear models

If you have a background in linear models, then much of the material will be familiar. If you're a statistician, obviously we're leaving out a lot of the 'details' (to say the least). Our purpose is to provide a minimum level of background, so even newcomers to linear models have a 'feel' for the approach. If you are new to linear models, we strongly suggest you supplement your reading of this chapter by having a look at one of the many good textbooks on this subject. McCullagh & Nelder (1989) and Dobson & Barnett (2008) are particularly good.

The basic idea underlying linear models can be stated quite simply: the response variable in many statistical analyses can be expressed as a linear regression function of 1 or more other factors. In fact, any ANOVA-type design can be analyzed using linear regression models (although interpretation of interactions is sometimes complex). In general, for data collected from marked individuals, the 'response variable' is often a probability or proportion (e.g., survival or recapture rate), which must be transformed prior to analysis using a linear models approach (we'll get to that in a moment). For the moment, assume the response variable has been suitably transformed.

We begin by demonstrating this relationship between 'regression' and 'ANOVA' by means of a simple

example. Consider data from a study where the skull circumference of young pre-school children is measured, and we're interested in knowing if this structure is on average larger in males than in females (we'll assume for the moment that all of the children were the same chronological age). Let's suppose we measure 7 male and 7 female children, and analyze our data using a normal single-classification ANOVA. Here are the data:

```
  male   7.2  7.1  9.1  7.2  7.3  7.2  7.5
female   9.8  8.5  8.7  8.6  8.4  7.7  8.2
```

First, the results from a 'standard ANOVA' (as you might generate using some statistical analysis software):

```
Source   df   SS     MS     F     P
SEX      1    3.806  3.806  8.33  0.0137
Error    12   5.485  0.457
Total    13   9.292
```

The results of this analysis indicate a marginally significant difference between male and female children.

However, what if our statistics package was limited only to a regression subroutine? Could we have analyzed our data using a linear regression model, instead of ANOVA, and arrived at the same result? The answer is, indeed, yes, we can. What we do is simply take the classification factor (`SEX`) and 'code' it as a '`0`' or '`1`' dummy variable (we'll see why in just a moment). For example, let '`0`' represent females, and '`1`' represent males. Thus, every individual in our data set is assigned a '`0`' or a '`1`', depending upon their gender. Let's call this dummy variable `SEX`. Now, all we need to do is regress our response variable (the skull circumference) on the dummy variable for `SEX`. Here are the results of the regression analysis:

```
Source   df   SS     MS     F     P
SEX      1    3.806  3.806  8.33  0.0137
Error    12   5.485  0.457
Total    13   9.292
```

No, it's not a typo – it is in fact the exact same table as above. The two approaches are entirely synonymous, yielding identical results. How can this be? The answer lies in the structure of the models actually being tested. So, let's step back to the beginning, and look at things a bit more formally.

In general, a linear model can be expressed in matrix form as

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}$$

where $\mathbf{y}$ is a vector of responses (i.e., a vector of the response variables), $\boldsymbol{\beta}$ is a vector of parameters (e.g., the intercept and 1 or more 'slopes'), $\mathbf{X}$ is a matrix with either '`0`' or '`1`' elements, or values of 'independent' variables, and $\boldsymbol{\epsilon}$ is a vector of random error terms.

In cases of analysis of variation of the response variable among different levels of one or more classification (i.e., 'treatment' or 'factor') levels, there is a parameter β in the vector $\boldsymbol{\beta}$ to represent each level of a factor. The elements of $\mathbf{X}$ (which is generally referred to as the *design matrix* – discussed below) are chosen to exclude or include the appropriate parameters for each observation. These elements are often referred to as either 'dummy' or 'indicator' variables ('indicator' generally being used when only '`1`' or '`0`' are used as the coding variables).

The following simple example will make this clear, and will illustrate the underlying connection between a linear regression model and analysis of variation (ANOVA). Suppose you have collected data on the scutum width of male and female individuals of some insect species. You are interested in whether or not the difference in mean scutum width between the sexes differs more than would be expected by random chance. Normally, you might consider using a single-classification (Model I) ANOVA for this sort of analysis. Recall that for this sort of analysis, any single variate Y (in this case, Y = scutum width), can be decomposed as:

$$Y_{ij} = \mu + \alpha_i + \epsilon_{ij}$$

In other words, each individual variate Y_{ij} is the sum of the global mean (μ), the deviation of the individual from that mean due to the 'classification' factor (sex; α_i), and the random error term (e_{ij}) In this example, with 2 levels of the classification factor (i.e., males and females), we would be testing for differences of the type ($\alpha_1 - \alpha_2$). If ($\alpha_1 - \alpha_2$) = 0 (the null hypothesis), then we would conclude no significant group effect (i.e., no significant difference in group means between the sexes).

How could we use linear regression to approach the same analysis? In a regression analysis, each individual variate Y_i would be decomposed as:

$$Y_i = \beta_1 + \beta_2 x_i + \epsilon_i$$

In this case, each variate Y_i is the sum of the product of the slope (β_2) and the variable x, the intercept (β_1), and a random error term (ϵ). In this case, the hypothesis being tested is whether or not the estimate of the slope is significantly different from 0 ($\mathbf{H_o}$: $\beta_2 = 0$).

However, what is the variable 'x'? In fact, this is the key to understanding the connection between the regression model and the ANOVA analysis. In the regression formulation, x represents a coding ('dummy') variable specifying male or female (i.e., sex, the classification variable in the ANOVA analysis). The coding variable takes on the value of '0' or '1' ('0' for females, '1' for males). We regress the response variable Y (scutum width) on the coding variable for sex. If the slope (β_2) is not different from 0, then we interpret this as evidence that the numerical value of the coding variable does not significantly influence variation in our data. Put another way, if the slope does not differ from 0, then this indicates no significant difference between the sexes. This is entirely analogous to test of the ($\alpha_1 - \alpha_2$) hypothesis in the ANOVA analysis.

Recall that we can express a linear model in matrix form as

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}$$

where $\mathbf{y}$ is a vector of responses (i.e., a vector of the response variables), $\boldsymbol{\beta}$ is a vector of parameters (e.g., the intercept and 1 or more 'slopes'), $\mathbf{X}$ is a matrix with either '0' or '1' elements, or values of 'independent' variables, and $\boldsymbol{\epsilon}$ is a vector of random error terms. For our present example, the design matrix $\mathbf{X}$ consists of 2 columns of '0' and '1' dummy variables (the first column corresponding to the intercept, β_1, and the second column corresponding to dummy variable coding for a given sex, β_2).

Given K individuals in each sex (although a balanced design is not required), $\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}$ can be written as

$$\begin{bmatrix} Y_{11} \\ Y_{12} \\ \vdots \\ Y_{1K} \\ Y_{21} \\ Y_{22} \\ \vdots \\ Y_{2K} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ \vdots & \vdots \\ 1 & 0 \\ 1 & 1 \\ 1 & 1 \\ \vdots & \vdots \\ 1 & 1 \end{bmatrix} \begin{bmatrix} \beta_1 \\ \beta_2 \end{bmatrix} + \begin{bmatrix} \epsilon_{11} \\ \epsilon_{12} \\ \vdots \\ \epsilon_{1K} \\ \epsilon_{21} \\ \epsilon_{22} \\ \vdots \\ \epsilon_{2K} \end{bmatrix}$$

In fact, in this case, if we used '1' to code for males, and '0' to code for females, then the intercept (β_1) would represent the estimate for female survival (since if the dummy variable is '0', then all that remains in the model is the intercept, and the random error term). The β_2 term actually reflects (male survival - female survival), such that $\beta_1 + \beta_2$ = (female) + (male-female) = male survival. The structure of the design matrix is discussed in more detail in the next section.

It is perhaps worth noting that models of the form '$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}$' are called linear models because the non-error part of the expression $\mathbf{X}\boldsymbol{\beta}$ is a *linear* combination of the parameters (and not specifically because of the relationship of ANOVA to linear regression). **MARK** uses this general linear models approach as the basis for all of the analysis (data) types available.

begin sidebar

matrix approach to linear regression & ANOVA: simple introduction

Here, we provide a *very* simple example of a matrix approach to linear regression (and, by extension, to linear models in general). For deeper understanding, you are strongly urged to consult one of the several very good textbooks which give *much* fuller treatments of the subject.

Consider the linear model, say of individual (i) with mass (Y_i) relative to sex (X_i, where $X = 0$ or $X = 1$ for – say – female or male, respectively), measured with Gaussian (normally) distributed random variation (ϵ_i) about the mean. We'll assume the following 'fake' data:

	mass (Y)			
male ($X = 1$)	11	12	11	14
female ($X = 0$)	8	11	12	10

The mean mass for males ($\bar{x}_m = 12$) is larger than the mean mass for females ($\bar{x}_f = 10.25$) – the usual question being, is the difference between the two larger than expected due to random chance?

We could adopt a linear models approach to answering this question – first, we could write the relationship between `mass` and `sex` in linear model form as

$$Y_i = \beta_1 + \beta_2 X_i + \epsilon_i$$

The null hypothesis of 'no difference between sexes' can be expressed formally in terms of the β term for sex; i.e., $\mathbf{H_o}: \beta_2 = 0$. The technical problem then is estimating the β_i coefficients in the linear model. To do this, first define a vector $\mathbf{y}$ for all the Y_i, a matrix $\mathbf{X}$ for a vector of 1s and all the X_i, a vector $\boldsymbol{\epsilon}$ for all the ϵ_i, and further define a vector $\boldsymbol{\beta}$ for the coefficients β_1 and β_2.

Then (for our 'fake' data set) we get

$$\mathbf{Y} = \begin{bmatrix} 11 \\ 12 \\ 11 \\ 14 \\ 8 \\ 11 \\ 12 \\ 10 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 1 \\ 1 & 1 \\ 1 & 1 \\ 1 & 0 \\ 1 & 0 \\ 1 & 0 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} \beta_1 \\ \beta_2 \end{bmatrix} + \begin{bmatrix} \epsilon_{11} \\ \epsilon_{12} \\ \epsilon_{13} \\ \epsilon_{14} \\ \epsilon_{21} \\ \epsilon_{22} \\ \epsilon_{23} \\ \epsilon_{24} \end{bmatrix} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}$$

Note that the matrix $\mathbf{X}$ is referred to as the *design matrix* – the construction of the design matrix is fundamental to using linear models in **MARK**, as we will cover in considerable detail later in this chapter. So, to derive estimates of the β_i coefficients, we need to find a vector $\boldsymbol{\beta}$ such that $\mathbf{y} = \mathbf{X}\boldsymbol{\beta}$. Is this possible? The answer is clearly 'no', because that would require the points to lie exactly on a straight line. A more modest (and tractable) question is: can we find a vector $\hat{\boldsymbol{\beta}}$ such that $\mathbf{X}\hat{\boldsymbol{\beta}}$ is in a sense 'as close to $\mathbf{y}$ as possible?'. The answer is 'yes'. The task is to find $\hat{\boldsymbol{\beta}}$ such that the length of the vector $\boldsymbol{\epsilon} = \mathbf{y} - \mathbf{X}\boldsymbol{\beta}$ is as small as possible (i.e., $\boldsymbol{\epsilon} \rightarrow 0$).

How do we get there from here? Fairly easily. First, we note that what we're trying to do is solve for $\boldsymbol{\beta}$ in the linear model. The first step is to let $\boldsymbol{\epsilon} = 0$ (such that it drops out of the equation – this should make sense, if you keep in mind that what we're trying to do is to find $\hat{\boldsymbol{\beta}}$ such that the length of the vector $\boldsymbol{\epsilon}$ is, in effect, 0). This leaves us with

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta}$$

Then, a few steps of algebra to solve for the vector $\boldsymbol{\beta}$:

$$\begin{aligned} \mathbf{y} &= \mathbf{X}\boldsymbol{\beta} \\ \mathbf{X}^{\mathbf{T}}\mathbf{y} &= \mathbf{X}^{\mathbf{T}}\mathbf{X}\boldsymbol{\beta} \\ \left(\mathbf{X}^{\mathbf{T}}\mathbf{X}\right)^{-1}\mathbf{X}^{\mathbf{T}}\mathbf{y} &= \left(\mathbf{X}^{\mathbf{T}}\mathbf{X}\right)^{-1}\mathbf{X}^{\mathbf{T}}\mathbf{X}\boldsymbol{\beta} \\ \left(\mathbf{X}^{\mathbf{T}}\mathbf{X}\right)^{-1}\mathbf{X}^{\mathbf{T}}\mathbf{y} &= \boldsymbol{\beta} \\ \hat{\boldsymbol{\beta}} &= \left(\mathbf{X}^{\mathbf{T}}\mathbf{X}\right)^{-1}\mathbf{X}^{\mathbf{T}}\mathbf{y} \end{aligned}$$

In words, we multiply both sides of the initial equation by the transpose of $\mathbf{X}$ to get the crossproduct $\mathbf{X}^{\mathbf{T}}\mathbf{X}$, which is a square matrix (*note*: the square matrix $(\mathbf{X}^{\mathbf{T}}\mathbf{X})$ is called the *pseudo inverse* of $\mathbf{X}$. We cannot use the true matrix inverse of $\mathbf{X}$ – that is, $\mathbf{X}^{-1}$ – because it generally does not exist as $\mathbf{X}$ is not a square matrix; $m \neq n$). We then find the inverse of this cross-product matrix and multiply both sides by that. This allows us to cancel out the term involving $\mathbf{X}$ on the right-hand side of the equation, allowing us to find an estimate of $\boldsymbol{\beta}$, which we call $\hat{\boldsymbol{\beta}}$, in terms of the original data.

It is worth noting that we could also approach this problem using the more familiar method of *least squares*. Recall that least squares involves minimizing the sum of the squared residuals between the observed and expected values. More formally, we want to minimize the Euclidean norm squared of the residual $(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})$, that is, the quantity

$$\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2 = \left([Y_1 - (\mathbf{X}\boldsymbol{\beta})_1]\right)^2 + \left([Y_2 - (\mathbf{X}\boldsymbol{\beta})_2]\right)^2 + \cdots + \left([Y_i - (\mathbf{X}\boldsymbol{\beta})_n]\right)^2$$

where $(\mathbf{X}\boldsymbol{\beta})_i$ denotes the ith component of the vector $(\mathbf{X}\boldsymbol{\beta})$.

We could also rewrite this as

$$\begin{aligned}\|\mathbf{y}-\mathbf{X}\boldsymbol{\beta}\|^2 &= ([Y_1-(\mathbf{X}\boldsymbol{\beta})_1])^2+([Y_2-(\mathbf{X}\boldsymbol{\beta})_2])^2+\cdots+[Y_i-(\mathbf{X}\boldsymbol{\beta})_n])^2\\ &= \sum_{i=1}^{n}(Y_i-(\beta_1+\beta_2 x_i))^2\end{aligned}$$

You might recall (from some linear algebra class you might have taken) that for some vector $\boldsymbol{\theta}$

$$\boldsymbol{\theta}^{\mathbf{T}}\boldsymbol{\theta}=\begin{bmatrix}\theta_1 & \theta_2 & \cdots & \theta_n\end{bmatrix}\begin{bmatrix}\theta_1\\ \theta_2\\ \vdots\\ \theta_n\end{bmatrix}=\theta_1^2+\theta_2^2+\cdots+\theta_n^2=\sum_i^n\theta_i^2$$

Thus, if $\boldsymbol{\theta}=(\mathbf{y}-\mathbf{X}\boldsymbol{\beta})$, then we can write

$$\begin{aligned}\|\mathbf{y}-\mathbf{X}\boldsymbol{\beta}\|^2 &= (\mathbf{y}-\mathbf{X}\boldsymbol{\beta})^{\mathbf{T}}(\mathbf{y}-\mathbf{X}\boldsymbol{\beta})\\ &= \mathbf{y}^{\mathbf{T}}\mathbf{y}-2\boldsymbol{\beta}^{\mathbf{T}}\mathbf{X}^{\mathbf{T}}\mathbf{y}+\boldsymbol{\beta}^{\mathbf{T}}\mathbf{X}^{\mathbf{T}}\mathbf{X}\boldsymbol{\beta}\end{aligned}$$

All that's left is to differentiate this expression with respect to $\boldsymbol{\beta}$, set to 0, and solve. Let

$$S=\|\mathbf{y}-\mathbf{X}\boldsymbol{\beta}\|^2=(\mathbf{y}-\mathbf{X}\boldsymbol{\beta})^{\mathbf{T}}(\mathbf{y}-\mathbf{X}\boldsymbol{\beta})$$

Thus,

$$\begin{aligned}\frac{\partial S}{\partial\boldsymbol{\beta}} &= -2\mathbf{X}^{\mathbf{T}}\mathbf{y}+2\mathbf{X}^{\mathbf{T}}\mathbf{X}\boldsymbol{\beta}=0\\ \mathbf{X}^{\mathbf{T}}\mathbf{Y} &= \mathbf{X}^{\mathbf{T}}\mathbf{X}\boldsymbol{\beta}\\ \hat{\boldsymbol{\beta}} &= (\mathbf{X}^{\mathbf{T}}\mathbf{X})^{-1}\mathbf{X}^{\mathbf{T}}\mathbf{y}\end{aligned}$$

Note this resulting algebraic solution is *identical* to that obtained earlier.

In fact, we could show that both solutions are equivalent to the MLE estimates for $\boldsymbol{\beta}$ (the Gaussian linear model is nice in the sense that the parameter estimates – namely the solution to the linear set of equations, the least squares estimate, and the maximum likelihood estimate – are all the same). For our 'fake' data:

$$\begin{aligned}\hat{\boldsymbol{\beta}} &= (\mathbf{X}^{\mathbf{T}}\mathbf{X})^{-1}\mathbf{X}^{\mathbf{T}}\mathbf{Y}\\ &= \begin{bmatrix}10.25\\ 1.75\end{bmatrix}\end{aligned}$$

Thus, our estimates for the intercept and slope are $\hat{\beta}_1=10.25$ and $\hat{\beta}_2=1.75$, respectively.

We would next estimate the error variance for $\hat{\beta}_1$ and $\hat{\beta}_2$. First, we derive an estimate of the variance-covariance matrix for the vector $\boldsymbol{\beta}$ estimates as

$$\text{var}(\hat{\boldsymbol{\beta}})=(\mathbf{X}^{\mathbf{T}}\mathbf{X})^{-1}\sigma_e^2$$

We can estimate σ_e^2 from the residual sums of squares (RSS) as

$$RSS=(\mathbf{y}-\mathbf{X}\boldsymbol{\beta})^{\mathbf{T}}(\mathbf{y}-\mathbf{X}\boldsymbol{\beta})$$

If the model estimates p parameters, then the estimate of σ_e^2 is simply $RSS/(N-p)$ where N is the number of data points. Thus,

$$\mathrm{var}(\hat{\boldsymbol{\beta}}) = \left(\mathbf{X}^{\mathbf{T}}\mathbf{X}\right)^{-1}(\mathbf{y}-\mathbf{X}\boldsymbol{\beta})^{\mathbf{T}}(\mathbf{y}-\mathbf{X}\boldsymbol{\beta})/(N-p)$$

So, for our 'fake' data (where $N = 8$ and $p = 2$), and our vector $\hat{\boldsymbol{\beta}}$,

$$\begin{aligned}\mathrm{RSS} &= (\mathbf{y}-\mathbf{X}\boldsymbol{\beta})^{\mathbf{T}}(\mathbf{y}-\mathbf{X}\boldsymbol{\beta}) \\ &= 14.75\end{aligned}$$

and thus

$$\begin{aligned}\mathrm{var}(\hat{\boldsymbol{\beta}}) &= (\mathbf{X}^{\mathbf{T}}\mathbf{X})^{-1}(\mathbf{y}-\mathbf{X}\boldsymbol{\beta})^{\mathbf{T}}(\mathbf{y}-\mathbf{X}\boldsymbol{\beta})/(N-p) \\ &= \begin{bmatrix} 0.6146 & -0.6146 \\ -0.6146 & 1.2292 \end{bmatrix}\end{aligned}$$

Thus $\widehat{\mathrm{SE}}(\hat{\beta}_1) = \sqrt{0.6146} = 0.7840$, and $\widehat{\mathrm{SE}}(\hat{\beta}_2) = \sqrt{1.2292} = 1.1087$. And, since a 95% CI for $\hat{\beta}_2$ (approximately $\hat{\beta}_2 \pm 2\mathrm{SE}$; $[-0.4674, 3.9674]$) clearly bounds 0, we would conclude no significant `sex` effect at a nominal $\alpha = 0.05$ level. Our 'hand calculated' estimates of slope and intercept, and variances for both parameters, are identical to the values returned by fitting the linear model in any statistical software package (below):

Variable	DF	Parameter Estimate	Standard Error	t Value	Pr > |t|
Intercept	1	10.25000	0.78395	13.07	<.0001
sex	1	1.75000	1.10868	1.58	0.1655

end sidebar

6.2. Linear models and the 'design matrix': the basics

In program **MARK**, the default design matrix for a given model is determined by the parameter structure of the model you are trying to fit (number of groups, and the number and structure of the parameters; i.e., the PIMs). This design matrix is then modified in various ways to examine the relative fit of different models to the data. In order to understand this process, it is essential that you understand how the design matrix is constructed.

We'll introduce the concept of a design matrix by means of an example. Suppose you are doing a 'typical' ANOVA on data with a single classification factor (say, 'treatment'). Suppose that there are 4 levels for this factor (perhaps a control, and 3 different levels of the 'treatment'). You want to test the hypothesis that there is no heterogeneity among 'treatment' levels ($\mathbf{H_o}$: $\mu_1 = \mu_2 = \mu_3 = \mu_4$). Recall from the preceding discussion that this problem can be formulated as an applied linear regression problem using '0/1 dummy variable' coding for the different levels of the 'treatment'.

Recall the previous example (above) which had 1 'treatment' or classification factor (`sex`), with 2 levels (male and female). The corresponding regression model was

$$Y_i = \beta_1 + \beta_2 x_i + \epsilon_i$$

where x represented a coding variable specifying male or female (i.e., `sex`, the classification variable in the ANOVA analysis). The coding variable took on the value of '`0`' or '`1`' ('`0`' for females, '`1`' for males).

What would the regression model look like for our present example, with 4 levels of the treatment factor instead of 2? How can we use a simple '`0`' or '`1`' dummy variable coding scheme (which clearly has only 2 'levels') to accommodate a treatment factor with 4 levels? The key is to consider the answer to the following question: if x_i can take on 1 of 2 values (0 or 1), then how many values of x_i do we need to specify k levels of the classification variable (i.e., the treatment variable)? If you think about it for a moment, you should realize that the answer is $k - 1$ (which, of course, corresponds to the degrees of freedom for a single-classification ANOVA).

Thus, for the present example, x_1, x_2 and x_3 could be:

$$x_1 = \begin{cases} \mathbf{1} \text{ if trt 1} \\ \mathbf{0} \text{ if other} \end{cases} \qquad x_2 = \begin{cases} \mathbf{1} \text{ if trt 2} \\ \mathbf{0} \text{ if other} \end{cases} \qquad x_3 = \begin{cases} \mathbf{1} \text{ if trt 3} \\ \mathbf{0} \text{ if other} \end{cases}$$

Clearly, when the coefficients for x_1, x_2 and x_3 are all 0, then the treatment level must be 4 ('other'). Thus, our regression equation for this example would be:

$$Y_i = \beta_1 + \beta_2 x_1 + +\beta_3 x_2 + \beta_4 x_3 + \epsilon_i$$

In this case, β_1 is the intercept, while β_2, β_3 and β_4 correspond to the slopes for each of the levels of the treatment factor. Since there are 4 levels of the treatment, 3 slopes are needed to code 4 levels of the treatment, because 1 of the levels of the treatment corresponds to the case where all 3 slopes are 0. Parameters β_2, β_3 and β_4 refer to treatment levels 1, 2, and 3, respectively. If $x_1 = x_2 = x_3$, then β_1 refers to treatment level 4. In other words, the intercept corresponds to treatment level 4.

begin sidebar

why is level 4 the intercept?

Choosing the intercept to specify treatment 4 was entirely arbitrary – we could for example have used any other level of the treatment as the intercept, and adjusted the coding for the remaining levels according. For example, we could have used used level 1 of the treatment as 'other' (i.e., the intercept), as follows:

$$x_1 = \begin{cases} \mathbf{1} \text{ if trt 2} \\ \mathbf{0} \text{ if other} \end{cases} \qquad x_2 = \begin{cases} \mathbf{1} \text{ if trt 3} \\ \mathbf{0} \text{ if other} \end{cases} \qquad x_3 = \begin{cases} \mathbf{1} \text{ if trt 4} \\ \mathbf{0} \text{ if other} \end{cases}$$

In this case, when the coefficients for x_1, x_2 and x_3 are all 0, then the treatment level must be 1 ('other'). Our regression equation would stay the same

$$Y_i = \beta_1 + \beta_2 x_1 + +\beta_3 x_2 + \beta_4 x_3 + \epsilon_i$$

but now, parameters β_2, β_3 and β_4 refer to treatment levels 2, 3, and 4, respectively. If $x_1 = x_2 = x_3$, then β_1 refers to treatment level 1.

What is important to note here is that in either case, one of the levels is specified by the intercept (i.e., β_1). This level is referred to as the '*control*' or '*reference*' level. In this design, then, the other levels ($\beta_2 \rightarrow \beta_4$) are 'offsets' from this reference (control) level (i.e., the other β terms represent the magnitude that a particular level of the treatment differs from the control). We will discuss this and related issues in much more detail later.

end sidebar

From this step, it is fairly straightforward to derive the design matrix (so-called because it fully represents the design of the analysis). The design matrix is simply a matrix showing the structure of

the 'dummy' coding variables in the analysis. Because there are 4 parameters being estimated in the equation (β_1, β_2, β_3 and β_4), each corresponding to the 4 levels of the main effect, then the design matrix will be a (4×4) square matrix.

To help construct the design matrix, we can decompose the general regression equation for this analysis (above) into n regression equations, where n is the number of parameters in the regression equation (i.e., the number of levels of the main effect; $n = 4$).

treatment	*equation*
1	$Y_i = \beta_1(\mathbf{1}) + \beta_2(\mathbf{1}) + \beta_3(\mathbf{0}) + \beta_4(\mathbf{0})$
2	$Y_i = \beta_1(\mathbf{1}) + \beta_2(\mathbf{0}) + \beta_3(\mathbf{1}) + \beta_4(\mathbf{0})$
3	$Y_i = \beta_1(\mathbf{1}) + \beta_2(\mathbf{0}) + \beta_3(\mathbf{0}) + \beta_4(\mathbf{1})$
4	$Y_i = \beta_1(\mathbf{1}) + \beta_2(\mathbf{0}) + \beta_3(\mathbf{0}) + \beta_4(\mathbf{0})$

The design matrix **X** simply corresponds to the matrix of the coefficient multipliers (in bold) in these equations.

$$\mathbf{X} = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

While this seems logical enough, there are, in fact, a number of alternative parameterizations of the design matrix, each of which yields the same 'model fit', but which have different interpretations.

For example, all 6 of the following design matrices ($\mathbf{X}_1$, $\mathbf{X}_2$ and $\mathbf{X}_3$) give equivalent model fits for our example problem:

$$\mathbf{X}_1 = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix} \quad \mathbf{X}_2 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{bmatrix} \quad \mathbf{X}_3 = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{X}_4 = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 \end{bmatrix} \quad \mathbf{X}_5 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \mathbf{X}_6 = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & -1 & -1 & -1 \end{bmatrix}$$

$\mathbf{X}_1$ (above) is the design matrix we derived previously; we estimate an intercept term for the last 'treatment' level (4), and then an additional 'treatment' effect for 'treatment' levels 1, 2 and 3. Matrices $\mathbf{X}_2 \rightarrow \mathbf{X}_4$ are based on the same underlying idea, except that the intercept specifies a different 'reference' level in each case (see preceding `-sidebar-`). For example, in $\mathbf{X}_2$, the intercept corresponds to treatment level 1. In $\mathbf{X}_3$, the intercept corresponds to treatment level 2. And, in $\mathbf{X}_4$, the intercept corresponds to treatment level 4.

In $\mathbf{X}_5$ is an *identity* design matrix. Here, each row corresponds to a parameter, and each column corresponds to a parameter. Thus, each parameter represents a treatment estimate directly, not as an 'offset' (deviation) from the 'control' or 'reference' (i.e., the intercept).

In $\mathbf{X}_6$, we estimate a mean parameter among treatment levels, and then an 'offset' for each of the 4 levels; the first column corresponds to the mean treatment value, and the remaining columns provide the treatment effects.

We'll consider these different design matrices later in the chapter. Note that the choice of the structure of the design matrix doesn't affect the estimates of the parameters (φ, or p, for example) – but it does change how estimates of the individual slope parameters in the linear model are interpreted. We will see many examples of this later in the chapter.

Perhaps the most important thing to remember in considering design matrices is that the number of rows corresponds to the number of parameters in your PIMs, whereas the number of columns corresponds to the number of these parameters you are trying to individually estimate. As we will see in the next section, this distinction becomes important when fitting models where parameters are constrained to be functions of 1 or more effects.

Finally, a more complex example, using 2 groups (say, males and females), with multiple levels of a treatment within group (i.e., within sex). This example is clearly analogous to a 2-way ANOVA, with 2 main 'effects' (treatment, and sex). Again, assume there are 4 possible treatment levels. The response variable Y can be decomposed as:

$$Y_{ijk} = \mu + \alpha_i + \beta_j + (\alpha\beta)_{ij} + \epsilon_{ijk}$$

where α_i is the sex (group) effect, β_j is the treatment effect, and $(\alpha\beta)_{ij}$ is the interaction of the two.

The corresponding regression equation would be:

$$\begin{aligned} Y_{ij} = {} & \beta_1 + \beta_2(\texttt{SEX}) + \beta_3(\texttt{t}_1) + \beta_4(\texttt{t}_2) + \beta_5(\texttt{t}_3) \\ & + \beta_6(\texttt{SEX.t}_1) + \beta_7(\texttt{SEX.t}_2) + \beta_8(\texttt{SEX.t}_3) + \epsilon \end{aligned}$$

If we derive the design matrix directly from this expression, then we see that we have 8 rows: 2 levels for `SEX` (male or female) multiplied by 4 treatment levels within sex (remember, $(n-1) = 3$ columns). The design matrix $\mathbf{X}$ (shown below) would also have 8 columns, corresponding to the intercept, the `SEX` (group effect), and the treatment and interaction terms, respectively

$$\mathbf{X} = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

The first column represents the intercept, the second column the group (`SEX`) effect (`1`=male, `0`=female; i.e., the additive effect of males-females), columns 3-5 represent the treatment effect ($t_1 \rightarrow t_3$), and columns 6-8 represent the interactions of `SEX` (male) and treatment. Why male, and not female? It depends on the coding – in this case, we're using '`0`' to represent females, and thus the interaction columns have non-zero elements for males only.

Suppose, for example, rather than the full model (with interactions), you wanted to fit the additive model consisting simply of the 2 main effects (no interaction term):

$$Y_{ijk} = \mu + \alpha_i + \beta_j + \epsilon_{ijk}$$

which, in regression form, is

$$Y_{ij} = \beta_1 + \beta_2(\mathtt{SEX}) + \beta_3(\mathtt{t}_1) + \beta_4(\mathtt{t}_2) + \beta_5(\mathtt{t}_3) + \epsilon$$

Using the design matrix **X** (above), this is easily accomplished by simply deleting the columns corresponding to the interaction terms:

$$\mathbf{X} = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Got it? As we work through this chapter, we'll come back to the concept of a 'linear model' and the 'design matrix' with considerable frequency, but hopefully you have the basic idea. In the examples we will explore in this chapter, you will learn the basic steps of creating these linear 'dummy variable' models, design matrices, and how to use them with **MARK** to test a variety of hypotheses.

The only thing we now need to consider is – how can we use 'regression models' for analysis of mark-recapture data, since both survival and recapture are not 'normal' response variables – normal in the sense that they are both constrained to be values from $0 \rightarrow 1$? If you simply regressed 'live=1/dead=0' or 'seen=1, not seen=0' on some set of explanatory variables **x**, it is quite conceivable that for some values of **x** the estimates value of survival or recapture would be > 1 or < 0, which clearly can't be correct! However, we clearly want to be able to bring the full power of ANOVA-type analyses to bear on capture-recapture studies.

As mentioned earlier in this chapter, the way around this problem is to transform the probability of survival or recapture, such that the transformed probabilities have been mapped from $[0, 1]$ to $[-\infty, +\infty]$, which is of course the 'assumption' for normal linear regression models. To accomplish this, **MARK** uses a link function (see the following 'sidebar' for more general background on link functions). In fact, **MARK** allows you to choose among a number of different link functions (some of which are more appropriate for certain types of analyses than others). The default link function is the sin link, which has very good properties for analyses that use what is known as the 'identity matrix' (much more on this matrix in a minute...). For models which don't use the identity matrix (such as constrained models), the logit link function is preferred (this is discussed later on in this chapter). Using these transformed probabilities, we can use linear regression models analogous to the one we just considered in the skull circumference example. We will now consider a simple example in detail, based on live encounter data from the European Dipper, to demonstrate how linear models are constructed using **MARK**.

begin sidebar

What is a link function?

In the context of analysis of data from marked individuals, a link function is a transformation of probability such that the transformed probability is mapped from $[0, 1]$ to $[-\infty, +\infty]$. For example, suppose you want to express a dichotomous (i.e., binary) response variable Y (e.g., survival or recapture) as a function of 1 or more explanatory variables. Let $Y = 1$ if alive or present; otherwise $Y = 0$. Let **x** be a vector of explanatory variables, and $p = \Pr(Y = 1 \mid x)$ is the probability of the response variable you want to model. We can construct a linear function of this probability by using a certain type of

transform of the probability, p. For example, the logit transformation (one of several transformation or link functions you can use with **MARK**) is given as:

$$\text{logit}(p) = \ln\left(\frac{p}{1-p}\right) = \beta_1 + \beta_2 x$$

where β_1 is the intercept, and $\boldsymbol{\beta}_2$ is the vector of slope parameters. Since $\theta = \ln(p/(1-p))$ has inverse $p = e^{\theta}/(1+e^{\theta}) = 1/(1+e^{-\theta})$, then the back-transformed estimate of $\hat{p}$ (i.e., back-transformed to the [0, 1] probability scale) is

$$\hat{p} = \frac{e^{\hat{\beta}_1 + \hat{\beta}_2 x}}{1 + e^{\hat{\beta}_1 + \hat{\beta}_2 x}} = \frac{1}{1 + e^{-\hat{\beta}_1 - \hat{\beta}_2 x}}$$

In other words, we can express the probability of the event (survival or recapture) as a linear function of a vector of explanatory variables. The logit (or logistic) model is a special case of a more general class of linear models where a function $f = f(m)$ of the mean of any arbitrary response variable is assumed to be linearly related to the vector of explanatory variables. The function f is the 'link' between the random component of the model (the response variable) and the fixed component (the explanatory variables). For this reason, the function $f(m)$ is often referred to as a 'link function'.

MARK allows you to choose among a number of different link functions (we will discuss the various link functions later in this chapter), some of which are more appropriate for certain types of analysis than others. **MARK** estimates the intercept and vector of the slope parameters, using the specified link, and then reconstitutes the values of the parameter from the values of the explanatory variables, x. **MARK** does this in 2 steps: (1) first, **MARK** reconstitutes estimates of the parameter from $\hat{\beta}_1$, $\hat{\beta}_2$ and x, and then (2) **MARK** computes values of the parameter from f using the back transform f^{-1}. There are several examples of this in the text.

end sidebar

6.3. The European Dipper – the effects of flooding

We return to our analysis of the European Dipper data set. Now, we will examine the effects of a specific climatic event (flood conditions on the breeding ground) on survival and recapture estimates.

As you may recall from Chapter 3 and Chapter 4, this data set involves 7 occasions of mark-recapture, of both male and female individuals. In those earlier chapters, we focussed on the males exclusively. In this chapter, we'll reanalyze this data set including both males and females. Each year in the study was characterized by the presence or absence of flood conditions. Are years with high (or low) values of either survival or recapture (or both) associated with years when there was a flood? Does the relationship between flood and either survival or recapture differ significantly between male and female Dippers? In order to address these questions, we will use the following 'logic sequence':

step 1 - *is there support for an interaction of sex and the covariate (flood) on variation in either survival or recapture?*

step 2 - *if there is no strong support for such an interaction, then is there evidence supporting a difference between the sexes in survival?*

step 3 - *in the absence of an interaction between sex and flood, is there any evidence supporting a linear relationship between survival (or recapture) and the covariate (flood)?*

This is the same sequence of steps used in analysis of covariance (ANCOVA). This is a very basic (and hopefully familiar) analytical design in statistical analysis, and we will demonstrate that the very same

approach can be used to analyze variation in survival or recapture. Before we begin, let's recast our analysis in terms of linear models. For the moment, let's use the simplified expression of linear models used in earlier chapters. Our basic model, including our classification variable (SEX) is

$$\varphi \text{ (or } p) = \texttt{SEX + FLOOD + SEX.FLOOD + error}$$

One thing you might ask at this point is – why isn't TIME included in the model? The answer lies in the fact that when we talk about constraints, we are speaking about 'applying' a constraint to a particular starting model. For example, we could start with the standard CJS model, with time-dependence of both survival and recapture probabilities. Then, we could apply a specific constraint to this model. In this example, we replace the 'random' effect of time in the CJS model by the 'specific' temporal effect of FLOOD. FLOOD is a particular case of time-dependence, because years with the same flood condition will share the same survival value. Thus, models with the FLOOD factor are 'nested' in the corresponding CJS model with the 'random' TIME factor. Of course, FLOOD, just like TIME, can be crossed (interaction) with SEX.

Our first step in this analysis is to test for the significance of the interaction term: SEX.FLOOD. If the interaction term is not significant, we can proceed to test for the significance of the other factors. How do we test for significance of the interaction term? Clearly, we test the model with the interaction against the same model without the interaction – using either relative model support (the QAIC approach), or (if you prefer) a LRT. The difference in fit of these two models is a 'test' of the interaction term.

$$\begin{array}{rl} & \varphi \text{ (or } p)\texttt{=SEX+FLOOD+SEX.FLOOD+error} \\ \textit{versus} & \varphi \text{ (or } p)\texttt{=SEX+FLOOD +error} \\ \hline & \texttt{SEX.FLOOD} \end{array}$$

How do we do this? The basic mechanics are the same as were described in earlier chapters – we use **MARK** to fit the 2 models we want to compare. But, in this case, there is a subtle difference; although the SEX term clearly corresponds to the 2 groups on our analysis, how do we incorporate the information specified by the FLOOD variable? In other words, how do we 'constrain' our estimates for either sex to be a linear function of flood? What about the design matrix? OK – here we go – step by step...

Step 1 – reading in the data

Start **MARK**, and create a new project. The data file for this example is the full Dipper data set (ED.INP). Select it, and label the 2 groups 'males' and 'females'. There are 7 occasions in the Dipper data set.

Step 2 – identify the parameters you want to constrain

In any typical analysis, your next step would be to decide on your starting, underlying model. For example, your starting model might include simple time-dependence in both parameters. Remember, this is the default starting model for **MARK**. How do you decide on the starting model? By using the techniques discussed in the preceding chapters, and the GOF procedures outlined in Chapter 5. Remember – you apply a constraint to a particular underlying (or starting) model – if this model doesn't adequately fit the data, then applying a constraint will not yield a particularly powerful test.

For the moment, let's assume that the model $\{\varphi_{g*t} p_{g*t}\}$ (i.e., time and group effects for both survival and recapture) is a good (and valid) starting model. Once you've determined the starting model, you need to determine the parameter indexing that **MARK** will use. For the Dipper data set, we have 2 groups, and 7 occasions.

Thus, the PIMs for this model would look like:

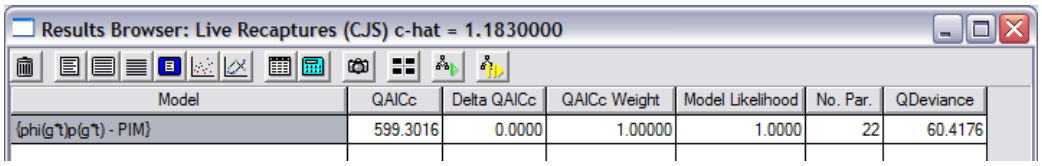
survival

1	2	3	4	5	6	7	8	9	10	11	12
	2	3	4	5	6		8	9	10	11	12
		3	4	5	6			9	10	11	12
			4	5	6				10	11	12
				5	6					11	12
males					6	*females*					12

recapture

13	14	15	16	17	18	19	20	21	22	23	24
	14	15	16	17	18		20	21	22	23	24
		15	16	17	18			21	22	23	24
			16	17	18				22	23	24
				17	18					23	24
males					18	*females*					24

Remember, this is the default model that **MARK** will start with, so there is no need to modify the PIMs at this stage. A GOF test (using the parametric bootstrap – see chapter 5) yields a $\hat{c}$ value of 1.183 (remember, your estimate of $\hat{c}$ might differ somewhat from this value, since the actual estimate of $\hat{c}$ will depend upon the number of bootstrap simulations you run). Adjust the $\hat{c}$ in the results browser from 1.000 (the default) to 1.183 (remember, this means we're now changing from AIC_c to $QAIC_c$). This model represents our 'starting point'. Note that there are 24 *structural* parameters – 6 survival parameters for each sex (12 total), and 6 recapture parameters for each sex (12 total). However, recall that there are a couple of non-estimable β-terms here, one for males and females, respectively, so 22 total *estimable* parameters (10 survival, 10 recapture, 2 β-terms). Go ahead and fit this model to the data. Call it '`phi(g*t)p(g*t) - PIM`' (we've added the PIM label so that when we look at the model in the browser, we'll know that this model was constructed using PIMs only).



Now, we'll fit a 'constrained model' to these data. For the moment, we're concentrating on survival in our analysis of these data, so the parameters we're interested in are in 'survival' matrices, the survival PIMs. Thus, we want to constrain 12 parameters: 6 survival estimates for males, and 6 for females. How do we do this? In other words, we want to make the probability of survival a linear function of other factors. In this particular example, the 'other factors' are `SEX`, and `FLOOD`.

Step 3 – defining the model structure of the linear constraint (modifying the design matrix)

As suggested earlier, linear models are specified via a 'design matrix'. In fact, this is precisely what we'll do in **MARK** – specify a particular design matrix, corresponding to the particular linear model we want to apply to the data. Again, recall that the name 'design matrix' reflects what it does – it is a

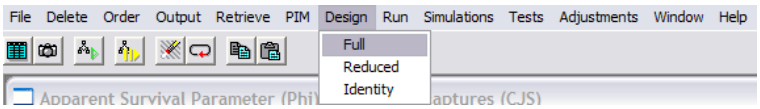
matrix containing the dummy variable structure which 'codes' the design of the linear model you are fitting to the data. What would the design matrix corresponding to model

$$\varphi \text{ (or } p) = \texttt{SEX} + \texttt{FLOOD} + \texttt{SEX.FLOOD} + \texttt{error}$$

look like? As a first step, we generally rewrite this model expression more formally. If we consider `FLOOD` as a simple binary variable (a year is either '`flood`' or '`non flood`'), then the corresponding linear model equation would be:

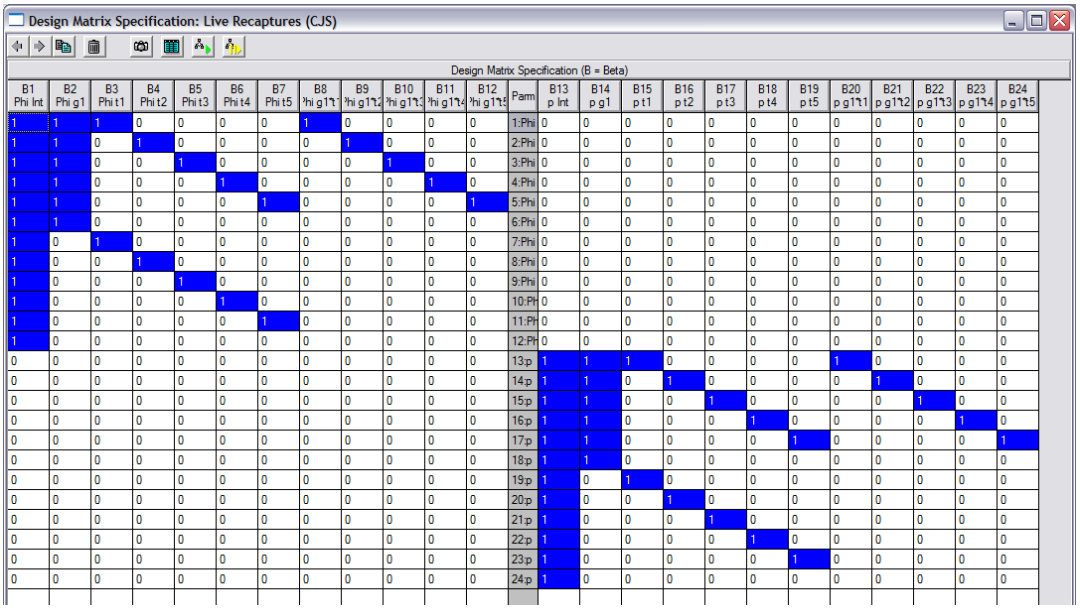
$$Y_{ij} = \beta_1 + \beta_2(\texttt{SEX}) + \beta_3(\texttt{FLOOD}) + \beta_4(\texttt{SEX.FLOOD}) + \epsilon$$

Recall that each column in the design matrix corresponds to each 'β' term in the model. In the model, we have 4 different 'β' terms – the intercept, the term for `SEX`, the term for `FLOOD`, and the term corresponding to the (`SEX.FLOOD`) interaction. So, the design matrix will have 4 columns. Now we need to create, or modify, the design matrix for this model in **MARK**. You may have noticed that there is a menu in **MARK** called '**`Design`**'. If you pull down the '**`Design`**' menu, you'll see that you are presented with several options: **`full`**, **`reduced`**, and **`identity`**.



Understanding the distinction between the various options in the '**`Design`**' menu will take a few steps, so for the moment, we'll start by selecting the '**`Design | Full`**' menu option from the '**`Design`**' menu.

Once you select '**`Full`**', a new window will pop up on the **MARK** desktop, looking like the following:

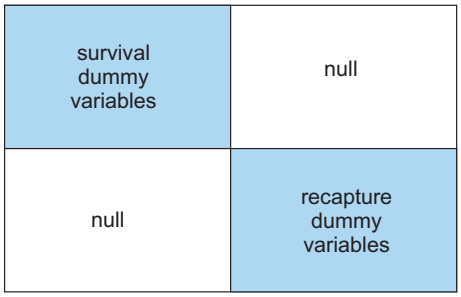


This new window is the '**`Design Matrix Specification`**'. [Here is where it helps to have a big monitor!] Here we are using the full Dipper data, consisting of 2 groups (males, females), and 7 sampling occasions). For larger data sets (more groups, more occasions), you may only see parts of it, so you need to get familiar with the basic layout. First, and most obviously, the matrix is split into a series of columns

In this example, there are in fact 25 columns, 1 each for each of the 24 'potentially' estimable parameters (remember, this is the CJS model, with 12 survival and 12 recapture parameters), and 1 'parameter label' column (the grey one in the middle of the matrix, labeled 'Parm'). At the top of each of the columns representing the 24 parameters, you'll see headers like 'B1', 'B2' and so forth. Recall that 'B' stands for 'β' – thus, the columns 'B1', 'B2', 'B3' refer to 'β_1', 'β_2' and 'β_3', respectively.

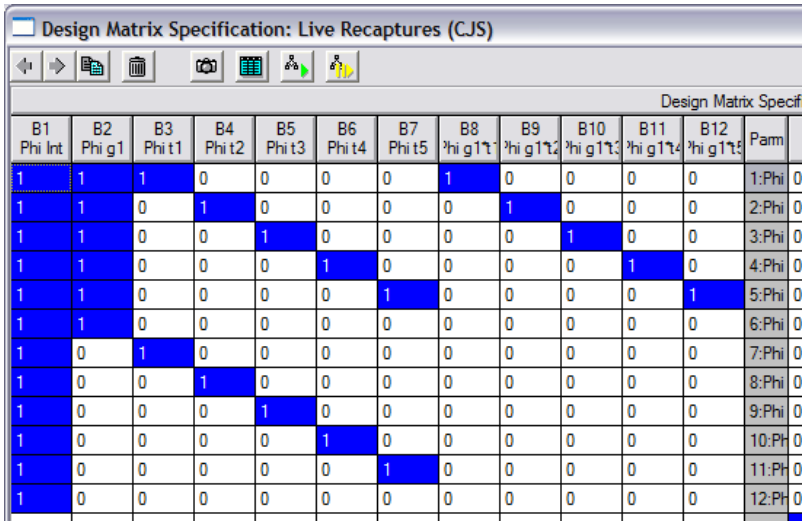
How many rows in the design matrix? You might guess 24. You would be correct! The design matrix – the full design matrix – for the CJS model for this data set is (in effect) a (24×24) matrix (we'll ignore the parameter column for the moment).

If you look at the design matrix for this model carefully, you'll see that it has the following general structure:



In the upper-left and lower-right quadrants, we have the 'dummy variable coding' for survival and recapture, respectively. The 'dummy variables' are shaded 'blue' when they're '1's (this is a convenience which you can turn off by flipping the right switch in the **MARK** preferences menu). In the upper-right and lower-left quadrants, we have what we'll refer to as the 'null' coding.

What are these codings? Let's look at the upper-left quadrant first – the dummy coding for the survival parameters. This quadrant is shown below:



First – how big (number of rows, number of columns) is the quadrant? Since the full matrix is (in effect) (24×24), then 1/4 of this is a (12×12) matrix. Thus, the upper-left quadrant are the first 12

columns and rows (going from left to right, top to bottom). In this quadrant (pictured on preceding page), we see a column of 12 '1's (the column labeled 'B1 - Phi Int'), a second column of 6 '1's followed by 6 '0's, then a series of columns with '1's going down along a diagonal. (*Note*: the column shows a label of 'B1 - Phi Int'. **MARK** provides the indicated column labels by default for the full design matrix corresponding to a fully time-dependent model structure, as specified by the PIMs. For other models, the default column labels are simply B*n* (e.g., B1, B2, B3...). To change the label for a particular column in the design matrix, simply right-click within any cell in a particular column. This will spawn a series of options you can select from – one of which is to label the column. Meaningful (to you) column labeling is very useful until you've developed some experience with design matrices – the labels will help you keep track of things.)

Remember, what is pictured here is the part of the design matrix which corresponds to the survival parameters for the 'full' general model - model $\{\varphi_{g*t}p_{g*t}\}$ (where g = SEX). In other words the basic CJS model for 2 groups. This is what '**Full**' refers to when you select that option from the '**Design**' menu - '**Full**' means the fully time-dependent model, as specified by the PIMs.

Second, the actual structure of this part of the design matrix reflects the linear model corresponding to model $\{\varphi_{g*t}\}$ (remember, we're only considering the survival part of the design matrix for now). Now, given that there are 7 occasions, and 2 groups (males and females), the linear model corresponding to model φ_{g*t} is (wait for it. . .)

$$\begin{aligned} Y_{ij} = {} & \beta_1 + \beta_2(\mathtt{SEX}) + \beta_3(\mathtt{t}_1) + \beta_4(\mathtt{t}_2) + \beta_5(\mathtt{t}_3) + \beta_6(\mathtt{t}_4) + \beta_7(\mathtt{t}_5) \\ & + \beta_8(\mathtt{SEX.t}_1) + \beta_9(\mathtt{SEX.t}_2) + \beta_{10}(\mathtt{SEX.t}_3) + \beta_{11}(\mathtt{SEX.t}_4) + \beta_{12}(\mathtt{SEX.t}_5) + \epsilon \end{aligned}$$

Pretty cumbersome looking, but not too hard if you go through it slowly. One term for the intercept (β_1), one term for the 'group' effect (i.e., sex, β_2), 5 terms for the 6 time intervals over which we hope to estimate survival (β_3 to β_7), and then 5 terms for the interaction of sex and time (β_8 to β_{12}) – a total of 12 parameters. Thus, 12 columns for this part of the design matrix, just as we observe in the preceding figure. Remember – even though there are 6 time intervals for survival (7 occasions = 6 intervals), we need only 5 columns – 5 parameters – to code them. So, for 6 intervals (which is analogous to 6 levels of a 'time' treatment), you need $(6 - 1) = 5$ parameters. This is precisely where the '$n - 1$' degrees of freedom bit comes from in ANOVA (and which is almost never explained to you in your basic stats class – they simply teach you 'rules' like '$n - 1$ degrees of freedom. . .' without explaining why. Now you know why! – it relates to how the linear model is set up and reflects the number of columns needed to code for a 'treatment' in the design matrix).

However, note that the columns of the design matrix are labeled 'B1' to 'B12'. **MARK** defaults to labeling the first column (the 'intercept' column) as 'B1', which seems counter to the (fairly) standard linear models convention of using β_0 for the intercept. While you can always change it in the **MARK** preferences if you want, we will adopt the convention of using β_1 for the intercept. Doing so makes it much easier to relate the terms of the linear model to specific columns in the design matrix (column B1 in the design matrix corresponds to β_1 in the linear model, column B2 corresponds to β_2, and so on).

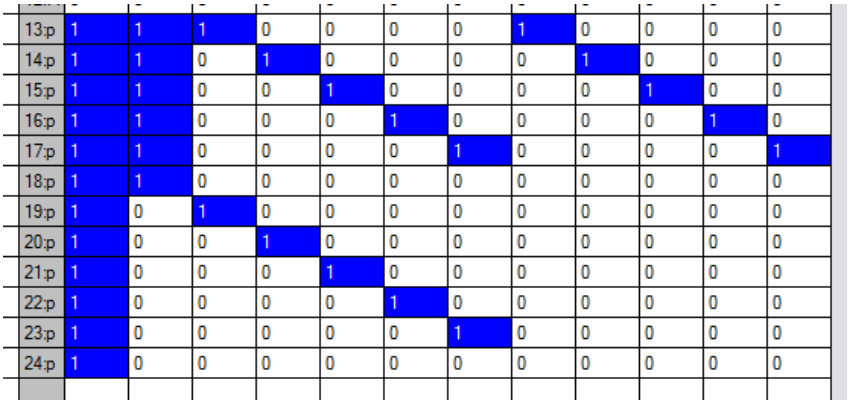
OK, so that's the basic structure – 12 columns, and 12 rows (12×12): 12 columns for the 12 parameters of the linear model, and 12 rows for the (2 groups × 6 occasions). Remember – the number of columns in the design matrix represents the number of parameters we want to estimate, while the number of rows in the design matrix reflects the number of parameters in the underlying model (i.e., the parameter indexing specified in the PIMs). If the number of columns is < the number of rows, then this implies that we are applying a *constraint* to the underlying model structure.

What about the actual dummy variable coding? Well, the intercept column should be straightforward – it's all '1's. Next – the column coding for 'SEX'. Here it is arbitrary which dummy variable you use to

code for 'males' and for 'females' (there is a fairly common convention for using '1's for males, and '0's for females, but it makes absolutely no difference at all). Again, there are 2 levels of this effect – 2 sexes. So, we need 1 column of dummy variables to code for sex (that old '$n - 1$ degrees of freedom' thing again). So far, so good – and hopefully, pretty easy stuff.

What about time? Well, if you remember the introduction to linear models and the design matrix from earlier in this chapter, you realize that what **MARK** does is use a row of all '0's to code for the last time interval, and then '1's along the diagonal to code for the preceding intervals. The choice of using '0's first, then the diagonal, is entirely arbitrary (you could, for example, use a diagonal set of '1's, with the last row being all '0's – makes little difference to the overall model fit, or the reconstituted estimates) – it is a **MARK** default. Note that this pattern is repeated twice – once for the males, and once for the females. Look closely – make sure you really do get it. Finally, the interaction terms. Pretty easy – just multiply the 'SEX' column by the 'TIME' columns to generate the 'SEX.TIME' interaction columns. Got it? Good!

Now, what about recaptures? So far, we've been talking only about modeling survival. Well, since the general model is $\{\varphi_{g*t} p_{g*t}\}$, then the structure for the design matrix for recaptures should be identical to the one for survival, except it is located in the lower right quadrant of the design matrix – the recapture part of the design matrix is pictured at the top of the next page. Before we proceed, what about the 2 'null' quadrants? Well, since 'null' generally refers to '0', you might suspect that the 'null' quadrants are filled entirely with '0's. You would be correct.



13:p	1	1	1	0	0	0	0	1	0	0	0	0
14:p	1	1	0	1	0	0	0	0	1	0	0	0
15:p	1	1	0	0	1	0	0	0	0	1	0	0
16:p	1	1	0	0	0	1	0	0	0	0	1	0
17:p	1	1	0	0	0	0	1	0	0	0	0	1
18:p	1	1	0	0	0	0	0	0	0	0	0	0
19:p	1	0	1	0	0	0	0	0	0	0	0	0
20:p	1	0	0	1	0	0	0	0	0	0	0	0
21:p	1	0	0	0	1	0	0	0	0	0	0	0
22:p	1	0	0	0	0	1	0	0	0	0	0	0
23:p	1	0	0	0	0	0	1	0	0	0	0	0
24:p	1	0	0	0	0	0	0	0	0	0	0	0

begin sidebar

changing the default reference level

In the preceding, we've used the default coding system in **MARK** to specify the '**full**' design matrix. The default uses the last time interval or occasion as the reference (i.e., the last time interval or occasion is represented by the intercept). As will be discussed later in this chapter (p. 77), there may be reasons why you don't want to use the last interval or occasion as the intercept – in particular, if it involves confounded parameters. For example, in a fully time-dependent CJS model, the final φ and p estimates are confounded. In such cases, it may make sense to change the default reference level to (say) the first interval or occasion. **MARK** makes it easy to do so – simply access '**File | Preferences**', and check the box '**For time effects in the using matrix, make the first row the reference value**'. Recall that the first row corresponds to the first interval or occasion.

end sidebar

Back to the problem at hand – we want to constrain the survival estimates – the first 12 parameters

(rows 1 → 6 for the males, and rows 7 → 12 for the females). In **MARK**, you constrain parameters by 'modifying' the design matrix. For our present example, we want to constrain survival to be a function of SEX, FLOOD, with a SEX.FLOOD interaction. Recapture probability we'll leave as is – with simple SEX and TIME differences, with a SEX.TIME interaction. Simple designs tend to be very easy, more complex designs obviously less so. So, we really need to consider only the structure of the design matrix for survival.

In fact, all you really need to do is write out the linear model equation for survival. In this case, it would be:

$$Y_{ij} = \beta_1 + \beta_2(\texttt{SEX}) + \beta_3(\texttt{FLOOD}) + \beta_4(\texttt{SEX.FLOOD}) + \epsilon$$

A term for the intercept (β_1), a term for the sex effect (β_2), a term for the flood effect (β_3 – remember that flood is a simple binary state variable – either 'flood' or 'no flood'), and a term for the interaction of the two (β_4). So, the survival part of the design matrix would consist of 4 columns, corresponding to this linear model. The number of rows? Again, the number of rows is the product of the number of time intervals specified in the PIMs, and the number of groups (so, $6 \times 2 = 12$ rows).

We already know how to code the intercept term, and the SEX term. What about the FLOOD? Well, since flood state is a binary variable, we can use '1' to indicate a flood year, and '0' to indicate no flood. In this study, the flood occurred during the 2nd and 3rd breeding seasons only.

Thus, the design matrix for the survival parameters will be – for males:

INTERCEPT	SEX	FLOOD	SEX.FLOOD
1	1	0	0
1	1	1	1
1	1	1	1
1	1	0	0
1	1	0	0
1	1	0	0

and for females

INTERCEPT	SEX	FLOOD	SEX.FLOOD
1	0	0	0
1	0	1	0
1	0	1	0
1	0	0	0
1	0	0	0
1	0	0	0

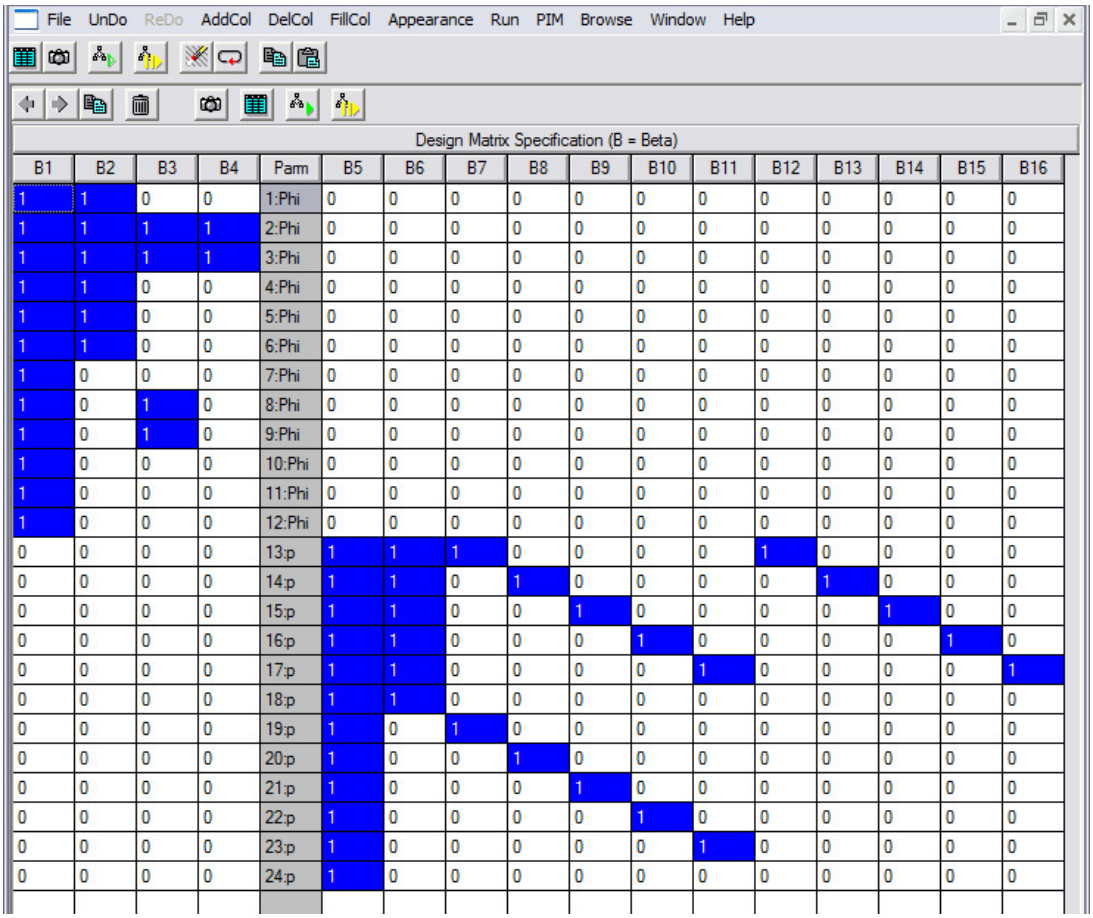
Note that since the SEX column is now all '0', the interaction column will also be a column of '0's, regardless of what is in the FLOOD column. Thus, putting the two sexes together, the design matrix for survival would be (top of the next page):

INTERCEPT	SEX	FLOOD	SEX.FLOOD
1	1	0	0
1	1	1	1
1	1	1	1
1	1	0	0
1	1	0	0
1	1	0	0
1	0	0	0
1	0	1	0
1	0	1	0
1	0	0	0
1	0	0	0
1	0	0	0

Got it? Now, all that's left is to translate this design matrix into **MARK**. There are a couple of ways to do this, but since we have the 'full design matrix' open already, we'll go ahead and modify it. Now, recall that in the unmodified full design matrix, we have 12 columns for the survival parameters, and 12 columns for the recapture parameters (we're ignoring the recapture parameters for the time being). So, we want to reduce the number of columns for the design matrix for survival from 12, to 4. It is this reduction that leads us to say that a model where survival is constrained to be a function of SEX and FLOOD is a 'reduced parameter model'. It is reduced because the number of columns (i.e., the number of parameters) is reduced, relative to the starting model.

Now, **MARK** makes it easy to manipulate the design matrix. But, for the moment, we'll do it 'manually', without some of the 'way cool and nifty' shortcuts that **MARK** offers. After some practice, you'll probably skip the manual approach, but then...the manual approach almost always works, even if it takes a bit longer. Basically, we want to keep the column corresponding to the intercept (the B1 column in the matrix **MARK** presents). We also want to keep the SEX column (column B2). Then, we want 2 columns: one for the flood dummy variable, and one for the interaction. The simplest approach to doing this is to manually edit the cells in columns 3 and 4 of the existing design matrix, entering the dummy variable coding for FLOOD, and the SEX.FLOOD interaction, as described earlier. All that's left after that is to delete the other 8 columns, which are no longer needed (there are lots of ways to delete or add columns to the design matrix – note the various menus specifically for this purpose. You can also right-click your way to the required structure). The final design matrix, showing both survival and recaptures, is shown at the top of the next page. Again, we now have 4 columns coding for the survival parameters, and 12 columns for the recapture parameters. Study this design matrix carefully – make sure you understand how it is constructed, and (obviously) why it has the structure it does.

Note: You may have noticed the grey-shaded column in the various design matrices we've examined thus far. This is a handy little feature of the presentation of the design matrix in **MARK**, which helps you remember which rows of the design matrix correspond to which parameters. This 'guide column' (for lack of a better name) can be dragged left or right within the design matrix – this is convenient since you can drag it to a point which conveniently separates the survival and recapture parameters to the left or right of the guide column, respectively (as we've shown in this example).



Design Matrix Specification (B = Beta)

B1	B2	B3	B4	Parm	B5	B6	B7	B8	B9	B10	B11	B12	B13	B14	B15	B16
1	1	0	0	1:Phi	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	2:Phi	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	3:Phi	0	0	0	0	0	0	0	0	0	0	0	0
1	1	0	0	4:Phi	0	0	0	0	0	0	0	0	0	0	0	0
1	1	0	0	5:Phi	0	0	0	0	0	0	0	0	0	0	0	0
1	1	0	0	6:Phi	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	7:Phi	0	0	0	0	0	0	0	0	0	0	0	0
1	0	1	0	8:Phi	0	0	0	0	0	0	0	0	0	0	0	0
1	0	1	0	9:Phi	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	10:Phi	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	11:Phi	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	12:Phi	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	13:p	1	1	1	0	0	0	0	1	0	0	0	0
0	0	0	0	14:p	1	1	0	1	0	0	0	0	1	0	0	0
0	0	0	0	15:p	1	1	0	0	1	0	0	0	0	1	0	0
0	0	0	0	16:p	1	1	0	0	0	1	0	0	0	0	1	0
0	0	0	0	17:p	1	1	0	0	0	0	1	0	0	0	0	1
0	0	0	0	18:p	1	1	0	0	0	0	0	0	0	0	0	0
0	0	0	0	19:p	1	0	1	0	0	0	0	0	0	0	0	0
0	0	0	0	20:p	1	0	0	1	0	0	0	0	0	0	0	0
0	0	0	0	21:p	1	0	0	0	1	0	0	0	0	0	0	0
0	0	0	0	22:p	1	0	0	0	0	1	0	0	0	0	0	0
0	0	0	0	23:p	1	0	0	0	0	0	1	0	0	0	0	0
0	0	0	0	24:p	1	0	0	0	0	0	0	0	0	0	0	0

6.3.1. Design matrix options: full, reduced, and identity

For the preceding example, we started by generating the 'full design matrix', corresponding to the general time-dependent model $\{\varphi_{g*t}p_{g*t}\}$. We did this by selecting the '**`Full`**' option from the '**`design matrix`**' menu. You might have noticed two other options – one for a '**`reduced`**' design matrix, and the other for an '**`identity`**' design matrix. The distinctions among the three options are:

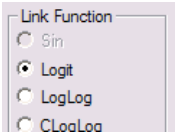
1. The '**`full`**' design matrix corresponds to a fully '`group` × `time`' model. If the underlying PIM structure is fully '`group` × `time`', then selecting the design matrix option '**`full`**' from the menu will generate the standard 'group × time' design matrix, based on intercept (reference) coding – **MARK** defaults to using the last time interval as the reference cell (this is discussed elsewhere). If you select the '**`full`**' design matrix option when the underlying PIM structure is not fully '`group` × `time`', then **MARK** will respond with an error box. The '**`full`**' design matrix option applies **only** if the underlying PIM structure, is fully 'group × time' – if the PIM structure is **anything** else, then you need to use the '**`reduced`**' option (see 2, below).
2. If either (i) the underlying PIM structure does not represent a fully '`group` × `time`' model (e.g., if the PIM structure represents a reduced parameter structure – fewer parameters than the '`group` × `time`' structure, or more parameters than a 'group × time' structure – for example, for a TSM model; Chapter 7), or (ii) you want to build a design matrix

which constrains the underlying PIM structure (e.g., applying a constraint to a 'group × time' PIM structure where time might be constrained to be a linear function of some environmental covariate), then you would select the '**reduced**' design matrix option (note: the term 'reduced' is perhaps somewhat misleading, since in fact you would use it for a model with more parameters than a '`group × time`' model – the word '**reduced**' would suggest fewer parameters). The 'reduced' option is the option you use whenever you want to construct a design matrix which does not correspond to a full '`group × time`' model. When you select '**reduced**', **MARK** presents you with a popup window which asks you to specify the number of columns you want in the design matrix, **MARK** defaults to the number of columns represented in the PIMs (i.e., if the PIMs specify n parameters, then the default design matrix will have n columns). If you want to start with fewer columns, you simply change the value in the popup window.

3. The '**identity**' matrix is the default design matrix in **MARK** – The Identity option results in the number of columns in the matrix equaling the number of rows, with the diagonal filled with ones and the rest of the matrix zeros. This is an identity matrix, and provides no constraints on the parameters. The identity matrix can be selected for any model, regardless of the underlying PIM structure.

6.4. Running the model: details of the output

Go ahead and run this model – call it '`Phi(sex*flood)p(sex*time)`'. Now, before you submit this model, something important to notice – the sin link is no longer the default.



In fact, as you can see (top of the next page), the sin link is not even available. The default (and generally preferred) link function when you change the design matrix from the default identity matrix is now the logit link.

begin sidebar

available link functions in MARK, and...why no sin link with a design matrix?

MARK currently supports 8 different link functions – 4 which constrain parameters to the interval $[0, 1]$ (logit, log-log, complementary log-log, and the default sin link), and 4 which either do not restrict the parameters to be in the $[0, 1]$ interval (identity and log), or which are constrained versions of the logit transform (cumulative and multinomial logit; these will be introduced later).

Although the logit function is commonly used for statistical analysis of $[0, 1]$ bounded parameters, this function presents some problems during the numerical optimization because as the parameter estimates for β approach $\pm\infty$ (i.e., as the real parameter values approach either 0 or 1), both the first and second derivatives of the function approach 0. As a result, determining the rank of the information matrix (see Chapter 4) is not a reliable method for determining the number of parameters estimated when using the logit link. In contrast, the sin link function is much better behaved numerically for the optimization – not only does this function require less computer time for the optimization (for some technical reasons), but because the second derivative is not zero when the value of the link function approaches either 0 or 1. As a result, the rank of the information matrix is a reliable estimator of the number of parameters estimated. Thus, the default link function in **MARK** is the sin link.

The following tabulates many of the more commonly used link functions and back-transformations in **MARK**, presented assuming a simple linear function $\theta = \beta_1 + \beta_2(x)$, where θ is some parameter bounded [0, 1] (e.g., encounter probability).

link	function	back-transform
sin	$\arcsin(2\theta - 1) = \beta_1 + \beta_2(x)$	$\theta = [\sin(\beta_1 + \beta_2(x)) + 1]/2$
logistic	$\log\left(\frac{\theta}{1-\theta}\right) = \beta_1 + \beta_2(x)$	$\theta = \frac{\exp^{(\beta_1+\beta_2(x))}}{1+\exp^{(\beta_1+\beta_2(x))}}$ (*)
log	$\ln(\theta) = \beta_1 + \beta_2(x)$	$\theta = \exp(\beta_1 + \beta_2(x))$
log-log	$\log(-\log(\theta)) = \beta_1 + \beta_2(x)$	$\theta = \exp(-\exp(\beta_1 + \beta_2(x))$
complementary log-log	$\log(-\log(1-\theta)) = \beta_1 + \beta_2(x)$	$\theta = 1 - \exp(-\exp(\beta_1 + \beta_2(x)))$
identity	$\theta = \beta_1 + \beta_2(x)$	$\theta = \beta_1 + \beta_2(x)$

*Note the equivalence of the following: $\frac{e^{X\beta}}{1+e^{X\beta}} = \frac{1}{1+e^{-X\beta}}$

Among the 6 standard link functions (excluding the cumulative and multinomial logit links), the log and identity link functions do not constrain the probability to the interval [0, 1], which can occasionally cause numerical problems when optimizing the likelihood. The log link does constrain real parameter values >1 (which is clearly useful for parameters which are naturally ≥ 0). For the log and identity link functions used with an identity design matrix, **MARK** uses the sin link function to obtain initial estimates for parameters, then transforms the estimates to the parameter space of the log and identity link functions when they are requested.

A common question – why is the sin link 'not available' when the design matrix is modified (i.e., when the design matrix is not an identity matrix)? As it turns out, the sin link should only be used with design matrices that are identity matrices, or when only one column in each row has a value not equal to zero, because the sin link will reflect around the parameter boundary, and not enforce monotonic relationships. In other words, despite some numerical advantages, the sin link function can produce biologically unrealistic models when the range of the covariate and/or the parameter estimates cause the link function to straddle 0. The logit link is better for non-identity design matrices.

The underlying reason has to do with the fact that the logit function is monotonic, whereas the sin function is not. Multiple values of the sin function will produce exactly the same transformation. For example, $sin(x)$, $sin(x + 2\pi)$, and $sin(x + 4\pi)$ all produce the same transformation. Such is not the case for the logit function. Thus, although the sin link is the best link function to enforce parameter values in the [0, 1] interval and yet obtain correct estimates of the number of parameters estimated, you need to be careful: in fact, because the sin link is not monotonic, the sin link is simply not available whenever you manipulate the design matrix (i.e., **MARK** is protecting you from making a possible mistake if you try to use the sin link).

Here is a worked example demonstrating the problem. Consider the survival of some organism deliberately exposed to an potentially toxic chemical. The data set (`sin_example.inp`) consists of 6 sampling occasions, and 2 groups (treatment and control). Analysis of this data set provides a very interesting example of the misbehavior of the sin link function. Fit the model $\{\varphi_{t+g} p_t\}$ with a sin link (which **MARK** will not let you do directly – in order to use a sin link here, you need to use a '**`parameter-specific`**' link, which we'll use now, but demonstrate later), and examine the parameter estimates (shown on the next page).

	Sin Link		Logit Link	
	Treatment	Control	Treatment	Control
Interval	Estimate	Estimate	Estimate	Estimate
1	0.693804	0.809271	0.723796	0.786824
2	0.966087	0.901265	0.913274	0.936837
3	0.902766	0.966995	0.921747	0.943152
4	0.908336	0.817021	0.846637	0.886047
5	0.684043	0.800919	0.700348	0.767003

The survival of the treatment and control groups are not consistent, i.e., compare the survival estimates from the sin link function in the following table. The control group is not consistently larger than the treatment group, or vice versa. Note how this constraint is enforced with the logit link function, i.e., the control estimate is always greater than the treatment estimate. The survival probabilities are parallel on the logit scale, but not on the sin scale. How can this be?

Again, the answer lies in the fact that the logit function is monotonic, whereas the sin function is not. Thus, the message is clear – beware when using the sin function in **MARK** for design matrices other than an identity matrix. You can get burned badly, as this example shows. The biological interpretation of the sin model is nonsense, yet, interestingly, this model provides the minimum AIC by 3 units. Further, the treatment effect would appear to be significant with this model. Because of the non-monotonic relationship provided by the sin link function, the sin link function is not easily available in **MARK** for design matrices where a row in the matrix has non-zero values in more than 1 column.

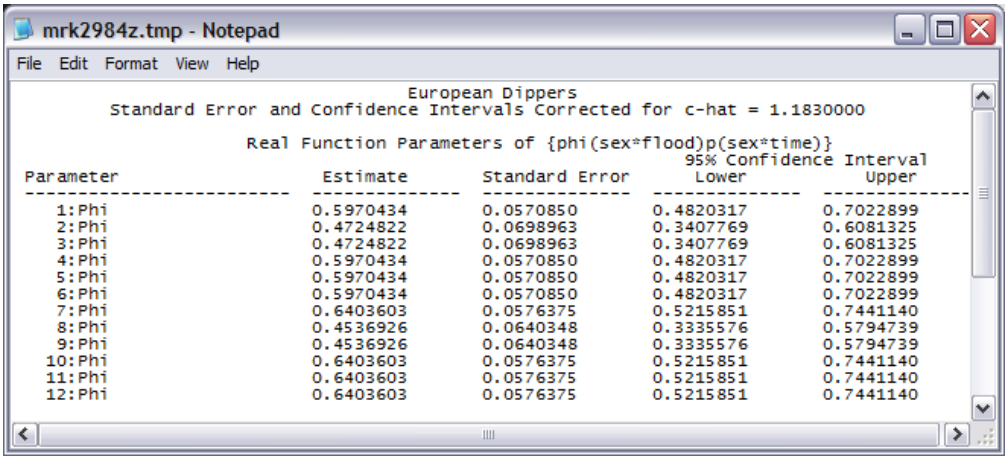
end sidebar

Go ahead and run this model, and add the results to the results browser. The $QAIC_c$ for this model (given the value for $\hat{c}$ we specified at the outset) is 584.77, which is approximately 15 less than our starting, general model. Thus, based on these 2 models, we would conclude that our 'constrained' (i.e., reduced parameter) model is **many** times better supported by the data than was our general starting model. However, note that our constrained model is reported to have 15 estimated parameters. And yet, if you look at the design matrix, you'll see that we have 16 columns, meaning 16 parameters.

So why does **MARK** only report 15, and not 16? In this case, the problem is with the data. **MARK** reports the numbers of parameters that it *can* estimate, given the data, not the number of parameters that are *theoretically* available to be estimated. If you look at the parameter estimates for the constrained model, you see that the recapture probability for males for the third occasion (reconstituted parameter 14) is estimated with a standard error of 0, which is usually diagnostic of there being a problem. As such, **MARK** doesn't 'count it' in the parameter total.

It is worth noting that one of the big differences between the logit link and the sin link is that the sin link generally does 'better' when dealing with parameters near the boundaries – meaning that it will often 'succeed' at estimating a few more of these 'problem' parameters than the logit link. However, you need to use the logit link if you modify the design matrix. So, keep it in mind. In this case, **MARK** reports only 15 parameters, not 16. So, you need to manually adjust the number of parameters to reflect the 'missing' parameter. You do this with the '**Adjustments**' menu. Simply change the number of parameters for this model from the 15 that are reported to the 16 that are structurally estimable.

Now, let's look at the estimates for survival, shown at the top of the next page. Couple of things to notice. First, the parameters $1 \rightarrow 6$ correspond to males, $7 \rightarrow 12$ to females. This is what we specified in the PIMs for the underlying model, to which you applied the constraint reflected in the linear model. Now, remember that there was a flood over the second and third intervals only. This is seen in the estimates – for males, for example, survival in the 2 flood years is 0.4725, while in the non-flood years, survival is 0.5970. For females, the survival during the 2 flood years is 0.4537, while for the non-flood years, it is 0.6403.



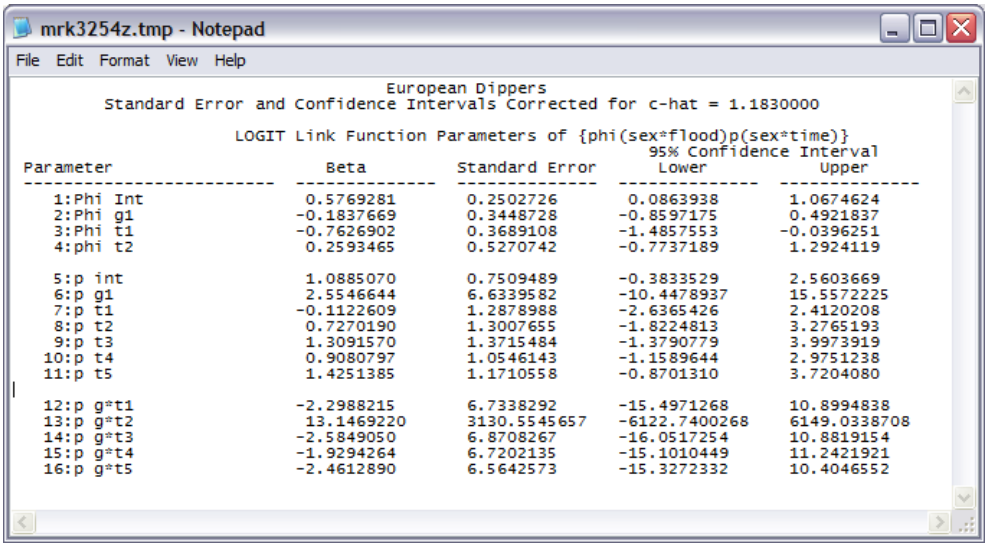
Where do these 'estimates' come from? Notice that these are labeled as '**Real Function Parameters**' – what does that mean? The answer is found if you look in the '**full output**'. Scroll down until you get past the section containing the details about the numerical estimation procedure. It is what comes after this 'summary' section which we're going to focus on for the moment. This next section is broken up into 2 distinct pieces: the **link function parameters** of the model, and the **real function parameters** of the model.

Very briefly, the '**link function parameters**' are the estimates of the 'slope parameters' in the linear model. Remember, we're fitting what is in effect a regression model to the data, a regression model that contains estimated parameters – these are the β values referred to earlier. In this example, they are logit link function parameter estimates. More about the linear model and the β-parameters in a moment. The real function parameters are the estimates of the survival and recapture parameters themselves. These values are estimated from the linear model. Think of it this way. The link function parameters define a linear equation, which is then used to estimate (or interpolate) the corresponding values of survival and recapture.

6.5. Reconstituting parameter values

Let's look at the link function parameter estimates a bit more closely. Recall that for the constrained model, 15 parameters were estimated. Can we confirm this by looking at the link function parameter estimates? You can look at the '**Beta**' estimates by clicking on model '`Phi(SEX*FLOOD)p(SEX*time)`' in the browser (to make it active), and then clicking on the third icon from the left in the browser toolbar. This will open up a notepad window (shown at the top of the next page) with the β-estimates (i.e., the estimates of the intercept and slopes).

We see that there are 16 β-estimates, but that only 15 of them are actually estimable (note the standard error for parameter 13). This is why **MARK** reports 15 estimable parameters. No confounded β-terms here (more on why in a minute). So we see again that the number of estimable parameters **MARK** reports in the browser corresponds to the number of estimable 'slopes' in the linear model.



But let's explore what these link function parameter estimates actually mean. This will make explicitly clear what link functions are all about. Let's consider the {SEX FLOOD SEX.FLOOD} model. The first 4 link function parameter estimates are:

Parameter	$\hat{\beta}$
1	0.576929
2	-0.183767
3	-0.762691
4	0.259347

These values are the coefficients ('slopes') for each of the terms in our linear model: one each for the INTERCEPT, SEX, FLOOD, and the SEX.FLOOD interaction (parameters $1 \rightarrow 4$, respectively). It is from these slopes and intercept (or, rather, from the linear model they define), that we 'reconstitute' our estimates for survival. A simple example will make this clear. Suppose you were given the equation $Y = 3.2x + 4$. If you were then given some value x, you could interpolate what the value of Y will be (on average, if the equation is a regression line). For example, if $x = 4$, then $Y = 16.8$. The same thing applies in our constraint analysis. We now have an equation of the form:

$$\text{logit}(\varphi) = \beta_1 + \beta_2(\texttt{SEX}) + \beta_3(\texttt{FLOOD}) + \beta_4(\texttt{SEX.FLOOD})$$

Now, from this equation, we can predict, or 'reconstitute' estimates of survival for any value of SEX, FLOOD and the interaction (SEX.FLOOD).

To make sure you really understand what is happening, let's consider how the reconstituted estimate for male survival over the fifth interval (i.e., $\hat{\varphi}_5$) is obtained. We must first compute the estimate of survival on the logit scale using the linear formula noted above, where the values of $\beta_1, \beta_2, \beta_3$ and β_4 are parameters ('beta') 1, 2, 3 and 4 (respectively). For males, SEX is '1' (this is our convention). As the fifth interval is a 'non-flood' year, FLOOD is '0', and thus the interaction term (SEX.FLOOD) is also '0'.

Therefore,

$$\begin{aligned}\text{logit}(\hat{\varphi}_5) &= 0.576929 + (-0.18377) \times (1) + (-0.76269) \times (0) + (0.25935) \times (0) \\ &= 0.393159\end{aligned}$$

The reciprocal of the logit transform is

$$\frac{e^{\text{logit}(\varphi_5)}}{1+e^{\text{logit}(\varphi_5)}}$$

Thus, the 'reconstituted' value of

$$\hat{\varphi}_5 = \frac{e^{0.393159}}{1+e^{0.393159}} = 0.5970429$$

This is the result given by **MARK** (up to the level of the rounding error).

begin sidebar

reconstituting standard error of estimate

In the preceding, we saw how we can 'back-transform' from the estimate of β on the logit scale to an estimate of some parameter θ (e.g., φ or p) on the probability scale (which is bounded on the interval $[0, 1]$). But, we're clearly also interested in an estimate of the variance (precision) of our estimate, on both scales. Your first thought might be to simply back-transform from the link function (in our example, the logit link), to the probability scale, just as we did above. But, does this work?

Consider the male Dipper data. Using the logit link, we fit model $\{\varphi_{\cdot}\, p_{\cdot}\}$ to the data – no time-dependence for either parameter. Let's consider only the estimate for $\hat{\varphi}$. The estimate for β for φ is 0.2648275. Thus, our estimate of $\hat{\varphi}$ on the probability scale is

$$\hat{\varphi} = \frac{e^{0.2648275}}{1+e^{0.2648275}} = \frac{1.303206}{2.303206} = 0.5658226$$

which is exactly what **MARK** reports (to within rounding error).

But, what about the variance? Well, if we look at the β estimates, **MARK** reports that the standard error for the estimate of β corresponding to survival is 0.1446688. If we simply back-transform this from the logit scale to the probability scale, we get

$$\hat{\varphi} = \frac{e^{0.1446688}}{1+e^{0.1446688}} = \frac{1.155657}{2.155657} = 0.5361043$$

However, **MARK** reports that the standard error for φ is 0.0355404, which isn't even remotely close to our back-transformed value of 0.5361043.

What has happened? Well, remember (from Chapter 1) that the variance for a parameter is estimated from the likelihood based on the rate of change in the likelihood at the MLE for that parameter (i.e., the second derivative of the likelihood evaluated at the MLE). As such, you can't simply back-transform from the SE on the logit scale to the probability scale, since the different scalings influence the shape of the likelihood surface, and thus the estimate of the SE. To get around this problem, we make use of something called the *Delta method*. The Delta method is particularly handy for calculating the variance of transformed variables (and clearly, that is what we are dealing with here). The details underlying the Delta method are beyond our scope at this point (the Delta method is treated more fully in Appendix B); here we simply demonstrate the application for the purpose of estimating the variance of the back-transformed parameter.

For example, suppose one has an MLE $\hat{\gamma}$ and an estimate of $\text{var}(\hat{\gamma})$, but makes the transformation,

$$\hat{\theta} = e^{\hat{\gamma}^2}$$

Then, using the Delta method, we can write

$$\widehat{\text{var}}(\hat{\theta}) \approx \left(\frac{\partial \hat{\theta}}{\partial \hat{\gamma}}\right)^2 \times \widehat{\text{var}}(\hat{\gamma})$$

So, all we need to do is differentiate the transformation function for θ with respect to γ, which yields $2\gamma.e^{\gamma^2}$. We would simply substitute this derivative into our expression for the variance, yielding

$$\widehat{\text{var}}(\hat{\theta}) \approx \left(2\hat{\gamma}.e^{\hat{\gamma}^2}\right)^2 \times \widehat{\text{var}}(\hat{\gamma})$$

Given values for $\hat{\gamma}$, and $\widehat{\text{var}}(\hat{\gamma})$, you could easily derive the estimate for $\widehat{\text{var}}(\hat{\theta})$.

What about for the logit transform? Actually, it's no more difficult, although the derivative is a bit 'uglier'. Since

$$\hat{\varphi} = \frac{e^{\hat{\beta}}}{1 + e^{\hat{\beta}}}$$

then

$$\begin{aligned}
\widehat{\text{var}}(\hat{\varphi}) &\approx \left(\frac{\partial \hat{\varphi}}{\partial \hat{\beta}}\right)^2 \times \widehat{\text{var}}(\hat{\beta}) \\
&= \left(\frac{e^{\hat{\beta}}}{1 + e^{\hat{\beta}}} - \frac{(e^{\hat{\beta}})^2}{1 + (e^{\hat{\beta}})^2}\right)^2 \times \widehat{\text{var}}(\hat{\beta}) \\
&= \left(\frac{e^{\hat{\beta}}}{(1 + e^{\hat{\beta}})^2}\right)^2 \times \widehat{\text{var}}(\hat{\beta})
\end{aligned}$$

It is worth noting that if

$$\hat{\varphi} = \frac{e^{\hat{\beta}}}{1 + e^{\hat{\beta}}}$$

then it can be easily shown that

$$\hat{\varphi}(1 - \hat{\varphi}) = \frac{e^{\hat{\beta}}}{\left(1 + e^{\hat{\beta}}\right)^2}$$

which is the derivative of φ with respect to β.

So, we could rewrite our expression for the variance of $\hat{\varphi}$ conveniently as

$$\begin{aligned}
\widehat{\text{var}}(\hat{\varphi}) &\approx \left(\frac{e^{\hat{\beta}}}{(1 + e^{\hat{\beta}})^2}\right)^2 \times \widehat{\text{var}}(\hat{\beta}) \\
&= (\hat{\varphi}(1 - \hat{\varphi}))^2 \times \widehat{\text{var}}(\hat{\beta})
\end{aligned}$$

From **MARK**, the estimate of the SE for $\hat{\beta}$ was 0.1446688. Thus, the estimate of $\hat{\text{var}}(\beta)$ is $(0.1446688)^2 = 0.02092906$. Given the estimate of $\hat{\beta}$ of 0.2648275, we substitute into the preceding expression, which yields

$$\widehat{\text{var}}(\hat{\varphi}) \approx \left(\frac{e^{\hat{\beta}}}{\left(1 + e^{\hat{\beta}}\right)^2}\right)^2 \times \widehat{\text{var}}(\hat{\beta})$$

$$= (0.0603525 \times 0.02092906)$$
$$= 0.001263$$

So, the estimated SE for $\hat{\varphi}$ is $\sqrt{0.001263} = 0.0355404$, which is what is reported by **MARK** (again, within rounding error).

Note: The standard approach to calculating 95% confidence limits for some parameter θ is $\theta \pm (1.96 \times \text{SE})$. However, to guarantee that the calculated 95% CI is [0, 1] bounded for parameters (like φ) that are [0, 1] bounded, **MARK** first calculates the 95% CI on the logit scale, before back-transforming to the real probability scale. However, because the logit transform is not linear, the *reconstituted* 95% CI will not be symmetrical around the parameter estimate, especially for parameters estimated near the [0, 1] boundaries.

end sidebar

We further distinguish between 'reconstituted' parameter estimates and 'free parameters' in the next section. In **MARK**, the output file does not distinguish between 'reconstituted parameters' and 'free parameters'. In fact, **MARK** arguably makes it a bit more difficult to see the correspondence between the number of constrained and free parameters. In **MARK**, all the parameters are printed in sequence – your only clue as to which are 'constrained' and which are 'free' is to look at the link function parameter estimates. This can be somewhat confusing for beginners. Recall that in this example, we applied the constraint to the survival estimates only. Thus, the recapture probabilities were estimated 'the normal way', although their value reflects the influence of the constrained survival estimates – this is why they are not the same as the estimates from the preceding CJS analysis. Got it?

If we follow the classical iterative modeling procedure discussed in earlier chapters, we might proceed to test reduced parameter versions of the 'flood model' by sequentially eliminating various terms in the model. For example, given the structure of the starting model, the first step would be to test for 'significance' of the interaction term. In other words, does the effect of flood on survival differ as a function of the sex of the organism? Recall that this is the prerequisite analysis in either multi-factorial ANOVA or ANCOVA.

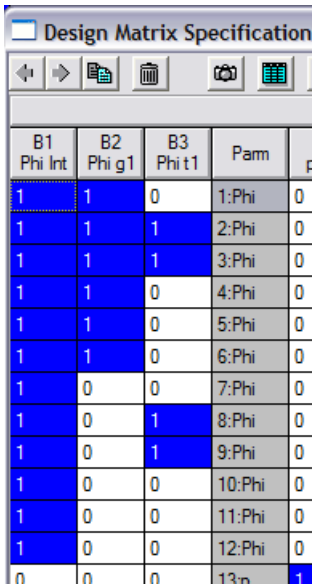
We would do this by comparing the following models:

	φ (or p)=SEX+FLOOD+SEX.FLOOD+error	
versus	φ (or p)=SEX+FLOOD +error	
		SEX.FLOOD

So, from top to bottom, we see that we first fit the 'full model', with both main effects and the interaction. We then follow this by fitting the reduced model without the interaction term. This model is what we refer to as an additive model – something we'll speak more about later in the chapter. The comparison of the fit of these models is a test of the significance of the interaction term. We've just finished fitting the full model, so we'll proceed directly to fitting the second model – without the interaction term. This analysis is very similar to what we've just done. All we need to do is modify the design matrix. Again, remember we are still applying the constraint to the underlying CJS time-dependent model. In fact, in **MARK**, this is very easy. All you need to do is drop the interaction term (in other words, delete the column containing the dummy variable coding for the interaction term – the fourth column, in our case) from the design matrix. That's it! Isn't that easy?

Go ahead and delete the interaction column from the design matrix, and then run this reduced model. Name it 'Phi(SEX+FLOOD)p(SEX*TIME)'. Note we change the syntax from 'SEX*FLOOD' to 'SEX+FLOOD' – a

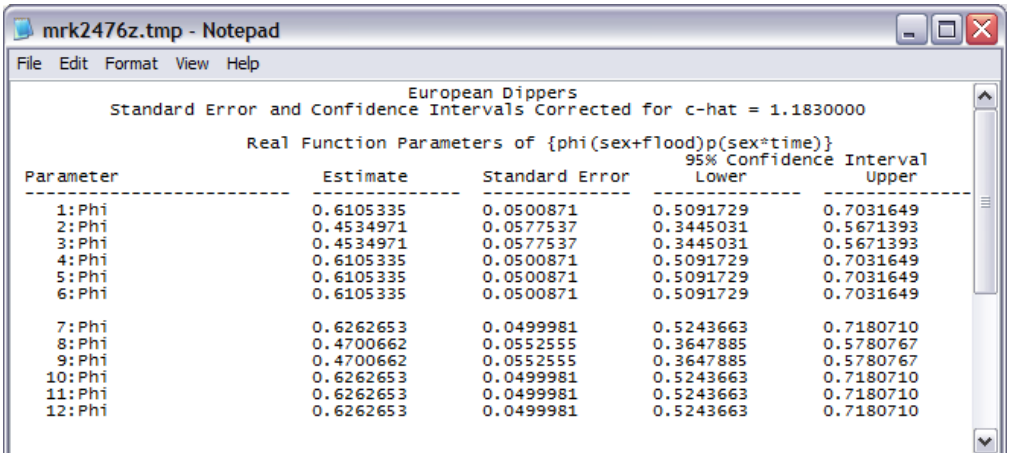
fairly standard convention to indication we've dropped the interaction term from the model. The newly modified design matrix (for the survival parameters) should now look like the figure shown below:



B1 Phi Int	B2 Phi g1	B3 Phi t1	Parm	
1	1	0	1:Phi	0
1	1	1	2:Phi	0
1	1	1	3:Phi	0
1	1	0	4:Phi	0
1	1	0	5:Phi	0
1	1	0	6:Phi	0
1	0	0	7:Phi	0
1	0	1	8:Phi	0
1	0	1	9:Phi	0
1	0	0	10:Phi	0
1	0	0	11:Phi	0
1	0	0	12:Phi	0
0	0	0	13:p	1

Look closely at this new model. First, instead of 3 slopes and 1 intercept, we now only have 2 slopes and one intercept. The slopes correspond to the `SEX` and `FLOOD` terms in our model, respectively. We have 1 fewer slope parameters since we eliminated the interaction term (`SEX.FLOOD`) from the model. Since we've dropped the interaction term, how many parameters should we have? Well, if we had 16 for the model with the interaction (remember – 15 were originally reported, but we manually adjusted this 'up' to 16 – see above), then we should have (at least in theory) 15 for the model without the interaction (since the interaction term corresponds to 1 link function parameter estimate). Note from the results browser that **MARK** reports 14 parameters – again, because of the fact that **MARK** was unable to correctly estimate p_3 for males, we need to adjust the number of parameters for this model 'up', from 14 to 15.

Next, we examine the 'reconstituted' survival estimates.



Again, we notice that there are effectively 2 estimates for each sex: one each for the flood or non-flood

years ($1 \rightarrow 6$ for males, $7 \rightarrow 12$ for females). Notice that the estimates are similar to, but not exactly the same, as the estimates with the full constraint (i.e., including the interaction term). In fact, on the logit scale, we see that the parameters parallel each other – with a constant difference between the males and females for a given year (again, on the logit scale).

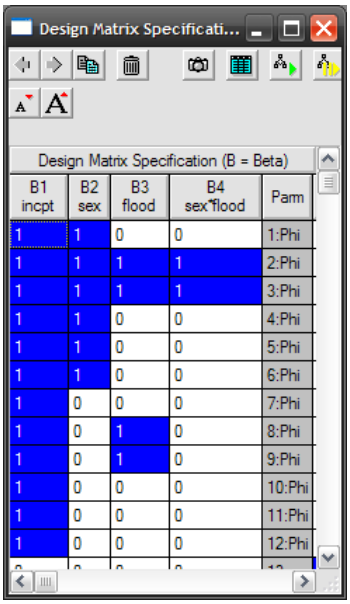
However, the key question is, is this difference 'large enough' to be 'biologically meaningful'? If we follow the 'classical' paradigm of model testing , we can compare the relative fits of the two models, keeping track of the number of parameters difference between the 2 models. From the results browser, we see that the $QAIC_c$ for the reduced model is 585.01, and for the full model, 586.93. Using the Akaike weights, the model without the interaction is approximately 2.6 times as well supported as the model with the interaction term – supporting the conclusion that there is no strong support for an interaction of `SEX` and `FLOOD`. The LRT results are consistent with this – the difference in deviance is not statistically significant by usual standards ($\chi^2_1 = 0.242, P > 0.5$). Since the interaction term is not significant, we could next proceed with testing the significance of the main effects: `SEX` and `FLOOD`. We can do so easily by using exactly the same process as we just completed above: we modify the constraint to include one of these two remaining terms, and compare the fit. However, while this allows us to test for significance of both terms, we must remember that we will not be able to use LRT to determine if the model containing `SEX` alone is a better model than one containing `FLOOD` alone. Why? Because these are not nested models. Thus, for comparison of these 2 models, we will have to use the $QAIC_c$ comparison approach. Even if the nesting isn't obvious (if it isn't, think about it! We will reconsider 'nestedness' in the context of linear models later in this chapter; see the `-sidebar-` beginning on p. 41), the necessity of using $QAIC_c$ for these 2 models will be obvious when you compare the number of parameters. In fact, this was one of the original motivations for use of the $QAIC_c$. However, as discussed in earlier chapters, the utility of $QAIC_c$ exceeds far beyond simple comparison of non-nested models.

If we were to proceed through each of the 'nested' models, we would see that the model where survival is constrained to be a function of `FLOOD` alone is the most parsimonious model, and is approximately 2.8 times better supported by the data than the next best model (`SEX+FLOOD`). No other model in the model set is adequately supported by the data. In other words, we conclude that there is no support for a 'significant' difference between the sexes, and that flooding significantly influences variation in survival. Does this mean that this is our best model overall? The answer to this question is 'no'. What we have done is simply to test a set of hypotheses under specific conditions. What were our main conditions? The conditions in this example were the use of the CJS model structure for both survival and recapture, prior to adding the constraints. The remaining question is – would we have come up with a different result if we had made the recapture probability constant? What if we had left time-dependence in recapture probability, but used the same parameter values between the sexes? How does our current model compare to one where survival is assumed to be constant?

Where we go from here, then, very much depends upon what we're after. We have to keep in mind the various purposes of model testing. At one level, we are seeking to test specific biological hypotheses. At the other, we may also be trying to find the most parsimonious model, which will provide us with the most precise and least biased estimates for modeling purposes. Again, our recommended strategy is to use the process of model selection to identify the most parsimonious acceptable model containing the effect(s) that you want to test, and then proceed to use LRT or $QAIC_c$ to compare this model with reduced parameter models where one or more of these terms have been eliminated. Remember, by 'acceptable' we mean a model which fits the data (Chapter 5). In fact, we can't emphasize this enough – the first step in analyzing your data must be to ensure that your starting model (CJS, for example) adequately fits the data.

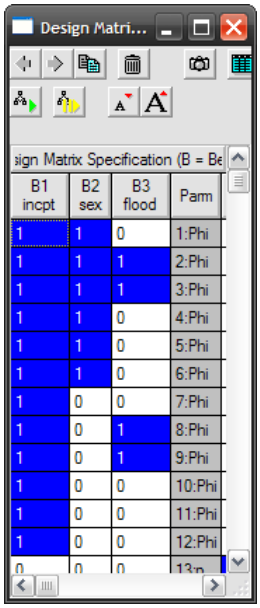
How would we derive the design matrix for the next 2 models – {`SEX`} and {`FLOOD`}? We can do this easily in **MARK**, using one of a couple of different approaches. The most intuitive, but perhaps laborious approach is to simply modify the design matrix manually.

For model {SEX+FLOOD}, we simply take the design matrix for {SEX+FLOOD+SEX.FLOOD},



B1 incpt	B2 sex	B3 flood	B4 sex*flood	Parm
1	1	0	0	1:Phi
1	1	1	1	2:Phi
1	1	1	1	3:Phi
1	1	0	0	4:Phi
1	1	0	0	5:Phi
1	1	0	0	6:Phi
1	0	0	0	7:Phi
1	0	1	0	8:Phi
1	0	1	0	9:Phi
1	0	0	0	10:Phi
1	0	0	0	11:Phi
1	0	0	0	12:Phi

and delete the column corresponding to the interaction term (shown at the top of the next page).



B1 incpt	B2 sex	B3 flood	Parm
1	1	0	1:Phi
1	1	1	2:Phi
1	1	1	3:Phi
1	1	0	4:Phi
1	1	0	5:Phi
1	1	0	6:Phi
1	0	0	7:Phi
1	0	1	8:Phi
1	0	1	9:Phi
1	0	0	10:Phi
1	0	0	11:Phi
1	0	0	12:Phi

Using the same approach, to fit model {SEX} all we'd need to do is take the design matrix for {SEX+FLOOD} and drop the FLOOD column, leaving {SEX}. But, once we've dropped the FLOOD column, how can we go from the design matrix for {SEX} back to {FLOOD}? Do we have to manually re-enter the appropriate dummy-variable coding? No! In **MARK**, all you need to do is click on any 'more saturated' model containing the terms you want in the results browser, and then 'retrieve' its design matrix. For example, if we want to fit model {FLOOD}, simply (a) click on the model {SEX+FLOOD} in the browser (this model is more saturated because it contains the factor of interest – FLOOD – plus one or more other

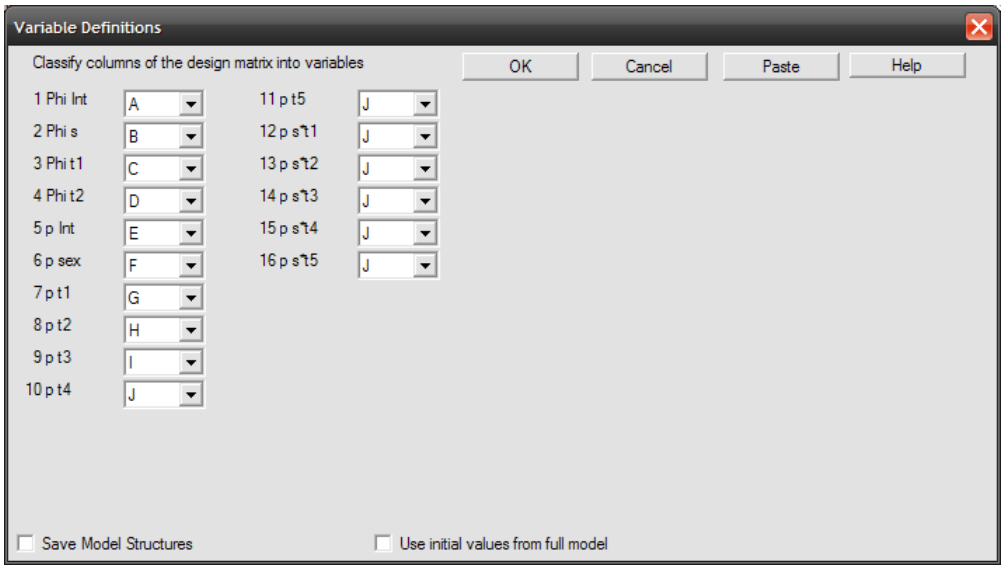
terms), then (b), pull down the '**Retrieve**' menu and retrieve the '**Current model**'. This causes **MARK** to extract the design matrix for the model you've selected. Once you have this design matrix (in this case, corresponding to model {SEX+FLOOD}), all you need to do is delete the SEX column to yield the design matrix for model {FLOOD}. Pretty slick!

6.5.1. Subset models and the design matrix

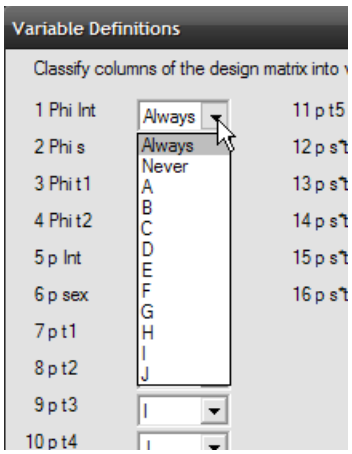
In the preceding example, we used the full design matrix from our general model $\{\varphi_{sex.flood}p_{sex.time}\}$, and built the reduced parameter models by deleting one or more columns from this design matrix. In this example, all of the remaining models in the candidate model set were nested within the general model.

While this is relatively straightforward to do 'by hand' for simple models, it can quickly become tedious for more complicated analysis, where the design matrices can be very large. **MARK** has an option referred to as 'subset models' to automate much of the process of constructing various nested models from the design matrix for some more general model. As the first step, you first construct the design matrix for the full model which contains all the variables (i.e., columns in the design matrix of interest). Note that the full model may not have actually been run, i.e., the saved structure of the full model can be used to construct the subset models. Then, using the 'subset models' feature, you simply select the columns from the full model design matrix to be used in a nested set of models.

To start the process, in the '**Results Browser Window**', highlight (retrieve) the model with the design matrix that contains all the variables that you want to use in all possible subset model combinations – in this case, model $\{\varphi_{s.f}p_{s.t}\}$. Then, select the menu choices '**Run | Subset of DM Models**' to create and run all possible models. This will bring up the interactive interface window shown at the top of the next page.



The interactive interface gives you a list of all the columns in the design matrix for the selected model. If you pull down any of the drop-down menus shown beside each parameter, you will see a set of options: '**Always**', '**Never**', followed by the letters **A** → **J**.



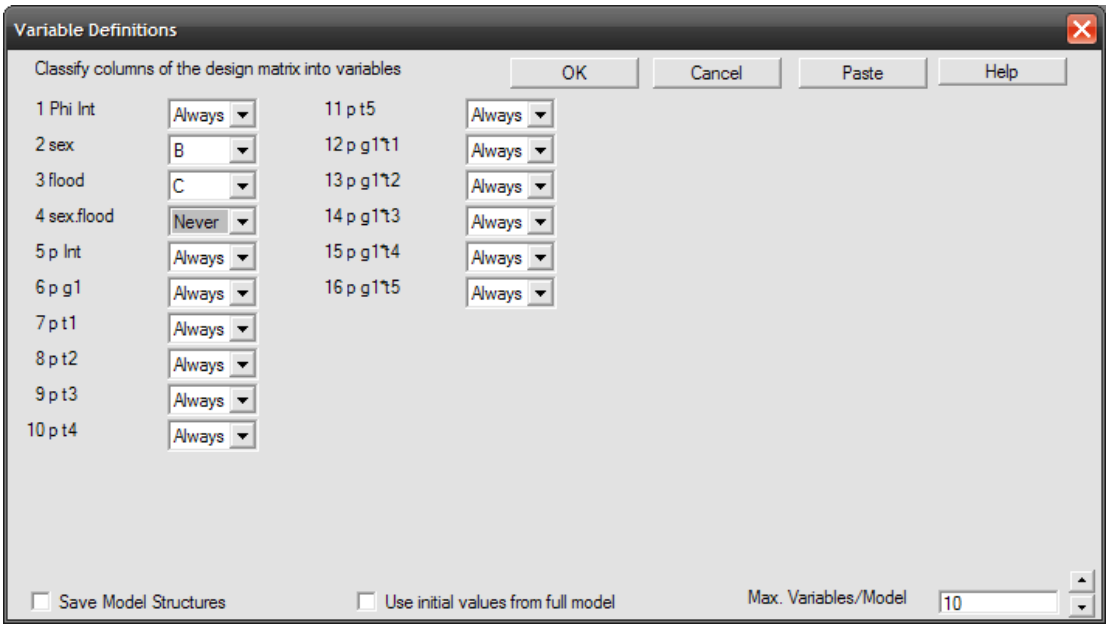
The 'subsetting' of different variables (columns) in the design matrix is accomplished by selecting elements from these drop-down lists for each parameter. Some of these options are relatively self-explanatory. For example, columns in the design matrix corresponding with a model 'intercept' (which we've designated in the label) we will generally keep in all models in the candidate model set, and so would be assigned the value '**Always**' to designate this status. For parameters (columns) which will never show up in any of the other models in the candidate model set, you would select the value '**Never**'.

The meaning and use of the values **A** → **J** requires a bit more explanation. Columns in the models which are neither '**always present**' or '**never present**' will be given values of **A**, **B**,...,**J**, designating up to 10 variables, either singly or jointly. Currently, the upper limit is set to 10 variables, which produces 2^{10} = 1,024 possible models. If you think you need to run more that 1,024 models, we suggest you think harder about the list of variables you are considering!

The letters **A** → **J** identify how columns in the design matrix are related to each other and how they will be combined in subsets of models. So, as an example, suppose as in the Dipper example you have 4 columns for apparent survival (φ) (the `intercept` column, the `sex` column, the `flood` column, and the (`sex.flood`) interaction column), and 12 columns for the encounter probability (p) (the `intercept` and `sex` columns, the 5 columns for time, and the 5 columns representing the interaction of (`sex.time`). In our candidate model set, we focus only on a series of 4 nested models for apparent survival: $\{\varphi_{sex+flood}\}, \{\varphi_{flood}\}, \{\varphi_{sex}\}, \{\varphi_{.}\}$. Recall that our general model is $\{\varphi_{sex.flood}\}$. Thus, the structure for our encounter probability $\{p_{sex.time}\}$ occurs in every model – in other words, it is 'always' there. So, as a first step, we simply select the value '**Always**' for the encounter parameters (columns).

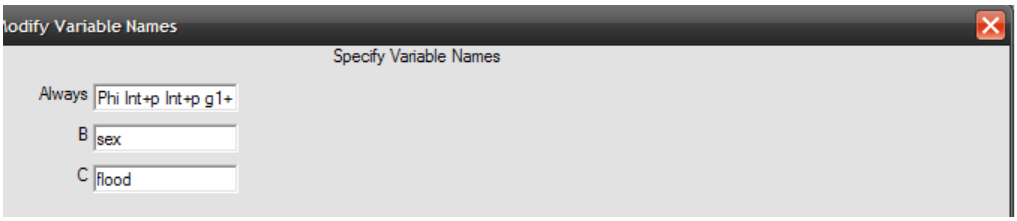
Now, for the apparent survival parameters. Clearly, the `intercept` is in each model, so we select the value '**Always**' for the intercept. The variables `sex` and `flood` are 'stand alone' variables – i.e., each defines a category with a single column in the design matrix. Each variable could appear alone in the model, so each is assigned its own letter. We'll use `A` for `sex`, and `B` for `flood` (it doesn't much matter which letters you use, so long as they are different). Now, what about the interaction term (`sex.flood`)? We need to use the value '**Never**', for two reasons: first, because the interaction model cannot enter the model without the two interacting variables also included in the model (i.e., model `Y=A+B+A.B` is valid, but `Y=A.B` is not), and moreover, there is no direct way to conditionally subset columns (i.e., select `C` if only `A` and `B` are present; discussed in more detail below). But second, and perhaps most obviously, we don't need the interaction column because that column is already present in the general model we started with. Think about it for a moment.

Here is what the variable definition screen should look like so far:



Now, before running the models, note the additional options which are available at the bottom of the model specification screen. Instead of running the models immediately, the model structure can be saved and then all of the models run later in batch model. The second option is to use the β estimates from the full model as initial values for the subset models. However, these estimates may not be great, depending on the collinearity among the variables. Note that this option does not appear if the full model has not actually been run (i.e., only the saved structure is used to specify the full model).

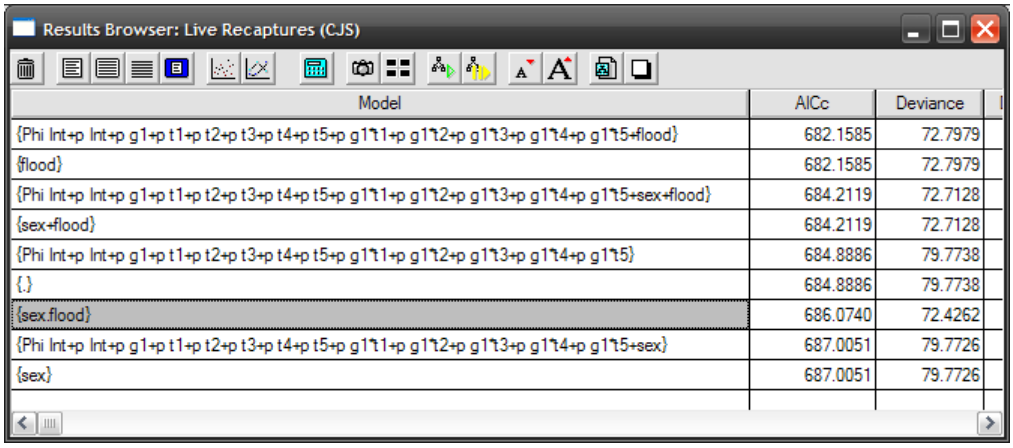
Once you hit the '**OK**' button, another window will pop up asking if you want to change (specify) variable names:



The naming of models fit using the '**subset models**' approach defaults to listing all of the columns which were 'always' included in the model. This can often result in very long model names (as we will see). This windows gives you the opportunity of overriding the default naming convention, but that places the burden of responsibility on you to remember which columns were included in your models.

Once you hit the '**OK**' button, a little popup window will inform you that you're going to run 4 models. But, before you simply click the '**OK**' button – think a bit. What are the 4 models? If you understand what we just did (above), then you'll know that the 4 models (in terms of φ) are $\{\varphi_{s+f}\}, \{\varphi_s\}, \{\varphi f\}, \{\varphi.\}$.

Here is the results browser, showing each of the models in the candidate model set fit by manually modifying the design matrix, and the 4 fit using the '**subset models**' approach, as well as the general model.



Model	AICc	Deviance
{Phi Int+p Int+p g1+p t1+p t2+p t3+p t4+p t5+p g1*t1+p g1*t2+p g1*t3+p g1*t4+p g1*t5+flood}	682.1585	72.7979
{flood}	682.1585	72.7979
{Phi Int+p Int+p g1+p t1+p t2+p t3+p t4+p t5+p g1*t1+p g1*t2+p g1*t3+p g1*t4+p g1*t5+sex+flood}	684.2119	72.7128
{sex+flood}	684.2119	72.7128
{Phi Int+p Int+p g1+p t1+p t2+p t3+p t4+p t5+p g1*t1+p g1*t2+p g1*t3+p g1*t4+p g1*t5}	684.8886	79.7738
{.}	684.8886	79.7738
{sex.flood}	686.0740	72.4262
{Phi Int+p Int+p g1+p t1+p t2+p t3+p t4+p t5+p g1*t1+p g1*t2+p g1*t3+p g1*t4+p g1*t5+sex}	687.0051	79.7726
{sex}	687.0051	79.7726

As noted earlier, the models fit using the '**subset models**' approach have very long model names – the naming syntax explicitly indicates which columns were included in the model. In particular, pay attention to the right-hand side of the default model names – this is where the 'variable' columns that are included in a given model are indicated. For example, model

```
{Phi Int+p Int+p g1+p t1+p t2+p t3+p t4+p t5+p g1*t1+p g1*t2+p g1*t3+p g1*t4+p
                                g1*t5+sex+flood}
```

The last two terms indicate that this model is the additive $\{\varphi_{s+f}\}$ model.

More importantly, notice that the model deviance values are identical for a given model regardless of whether or not it was generated by modifying the design matrix manually, or using the 'subset models' approach.

The preceding analysis was very simple, and the cost savings for using the 'subset models' approach might not be particularly significant in this instance. But, that will generally not be the case.

A common issue alluded to earlier occurs when you only want to include a particular column when another column is included in the model. An example would be for linear trend (T) and the associated quadratic trend (TT). As an example, suppose that there are two additional variables, age and gender. One approach to only including TT when T is in the model is to do 2 sets of models. For the first set, only the T variable would be used:

```
intercept   Always
age         A
gender      B
T           C
TT          Never
```

Then, a second set of models are constructed to always include TT with T:

```
intercept   Always
age         A
```

```
gender      B
T           C
TT          C
```

Each set would produce 8 models, for a total of 16. However, the user will have 4 sets of duplicates when neither T or TT are included:

```
intercept
intercept + age
intercept + gender
intercept + age + gender
```

Sorting (ordering) the list of models by model name may help find the duplicates.

A second issue is that the user never wants 2 particular variables in the model at the same time. Suppose this is the case for length and weight. Again, a simple solution is to run 2 sets of models, specifying the '**Never**' key word first for length, and then for weight. However, again, some duplicate models will have to be removed.

6.6. Some additional design matrix tricks

In a moment, we'll continue with the next 'analytical question' we might be interested in – are the differences between flood and non-flood significant? Is there an interaction of `flood`, `sex` and `survival`? And so on. For the moment, though, let's consider a couple of the 'mechanical' aspects of modifying the design matrix which are worth knowing. In the preceding, we modified the design matrix manually. As you probably realize by now, **MARK** often gives you more than one way to do things (this is generally a good thing!). What could we have done other than manually editing the design matrix. A perhaps more elegant, and (with some practice) faster way, is using some of **MARK**s nifty menu options. For example, pull down the '**FillCol**' menu. You'll see two intercept options – the '**Intercept**' itself, and something called the '**Partial Intercept**'. Since in our example we only wanted to modify 'part' of the design matrix, we would select '**Partial Intercept**'. This causes **MARK** to spawn a window asking you the number of rows in the design matrix you want to add the intercept coding to. In this example, with 12 rows (corresponding to the 12 survival parameters – 6 for males, 6 for females), we would have responded with '12'. Anything else we could do with the '**FillCol**' menu? Well, you might notice that there is an option to specify a '**Group Effect**' item on the menu. If you select this option, another child menu will pop up, giving you several options for the kind of group effect you want to code (broadly dichotomized into discrete or continuous). Now, within the discrete or continuous groupings, you'll see options for '**partial**'. What does this mean? Well, 'partial' simply means we want whatever it is we're going to do to be applied only to 'part' of the design matrix. Since in our example we're only interested in the first 12 parameters, corresponding to the first 12 rows and columns, we clearly would want 'partial' – a piece of the whole matrix. If we select '**Partial Discrete**', **MARK** would immediately fill in the first column of the design matrix, with 6 '1's followed by 6 '0's. **MARK** is clever enough to remember that (a) you have 2 groups, and (b) that there are 7 occasions (and therefore 6 parameters) for each group. In fact, it 'learned' this when you filled in the model specification window for this analysis. Pretty slick, eh? At any rate, go ahead and 'play around' a bit with the various menus available for modifying the design matrix.

One last thing – remember at the beginning of our example – we started with the '**Full**' design matrix? Recall that there were 2 other options in the '**Design Matrix**' menu - '**Reduced**', and '**Identity**'. The '**Reduced**' option allows you to tell **MARK** exactly how many columns to put into the design matrix –

this can be useful (and can save you some time) *if you know how many columns you need* – which you might if you've carefully thought through the linear model, and corresponding design matrix, for your analysis. The '**Identity**' matrix is simply a matrix of the same dimension as the '**Full**' design matrix, but with '1's along the diagonal. Some people prefer modifying the identity matrix, since most of the matrix elements are '0's – fewer cells to modify. Pick whichever approach works best for you.

6.7. Design matrix...or PIMs

In the preceding, we fit the 'flood model' by modifying the design matrix. Remember, the design matrix reflects the underlying structure of the model, which is specified by the PIMs. When we modify the design matrix, for a given set of PIMs, we're applying a constraint to the model specified by those PIMs.

This is a very important point – so important, that we're now going to force you to think about it carefully, by pointing out that we could have fit a 'flood' model to the Dipper data without using a linear model at all – simply by using a different set of PIMs! How? Recall that our original starting model was the fully time-dependent CJS model with two groups (the two sexes, males and females). There were 7 occasions in the data set, so the PIMs reflecting this starting model were

survival

1	2	3	4	5	6	7	8	9	10	11	12
	2	3	4	5	6		8	9	10	11	12
		3	4	5	6			9	10	11	12
			4	5	6				10	11	12
				5	6					11	12
males					6	*females*					12

recapture

13	14	15	16	17	18	19	20	21	22	23	24
	14	15	16	17	18		20	21	22	23	24
		15	16	17	18			21	22	23	24
			16	17	18				22	23	24
				17	18					23	24
males					18	*females*					24

These PIMs specified the model that we then constrained – they indicate 6 survival parameters for each sex, which we then constrained to be a linear function of flood. Remember, the actual linear model we fit initially was:

$$\text{logit}(\varphi) = \beta_1 + \beta_2 \texttt{SEX} + \beta_3 \texttt{FLOOD} + \beta_4 \texttt{SEX.FLOOD}$$

Now...is there any way we could have fit this model without modifying the design matrix? The answer is...'yes'. How? A couple of hints: PIM's, and the fact that flood is a binary state variable.

Remember, a year is classified as either a flood year, or a non-flood year. In our original survival PIMs, we had 6 parameters for each sex, respectively, which allowed survival to vary among years. However, if we want survival to vary only as a function of whether or not a year is a 'flood year', then in fact we need only 2 parameters for each sex!

Thus, we could have specified model

$$\text{logit}(\varphi) = \beta_1 + \beta_2\texttt{SEX} + \beta_3\texttt{FLOOD} + \beta_4\texttt{SEX.FLOOD}$$

using the following PIMs for survival:

survival

1	2	2	1	1	1
	2	2	1	1	1
		2	1	1	1
			1	1	1
				1	1
					1

males

3	4	4	3	3	3
	4	4	3	3	3
		4	3	3	3
			3	3	3
				3	3
					3

females

What would the recapture PIMs look like? Well, if we use model $\{p_{g*t}\}$ for recapture, the PIMs for the recapture parameters would look like:

recapture

5	6	7	8	9	10
	6	7	8	9	10
		7	8	9	10
			8	9	10
				9	10
					10

males

11	12	13	14	15	16
	12	13	14	15	16
		13	14	15	16
			14	15	16
				15	16
					16

females

Try it! You'll see that you get the same results as you did when you modified the design matrix – **IF** you specify the logit link function. In fact, one advantage of this approach is that you can use whatever link function you want. Note also that the PIMs specify 16 parameters, which is exactly the same number of parameters as there were in the design matrix approach (although only 15 of them were estimable using the logit link – if you run this PIM-based model using the sin link, all 16 parameters are estimated, and reported as estimated by **MARK**).

Pretty slick, eh? Make sure you understand this. If not, go through it again. Now, if this approach works, why not use it all the time? The reason is, because it has limited flexibility in the ability to specify certain kinds of models – you can't use this approach to build additive models (something we'll address later), or to constrain estimates to follow a trend (the next section). There are limits – but it is useful to know that you can sometimes 'get where you need to go' by modifying the PIMs directly, rather than using the design matrix. This is important both conceptually, and practically, in some cases. However, we generally recommend that you use the design matrix approach the majority of the time, since it gives you the greatest flexibility. In fact, in many cases, the design matrix is your only option – the additive (SEX+FLOOD) model is a good example of this. There is **no** way you could build this additive model using PIMs alone.

So, as a general recommendation, we suggest the following sequence (which we'll revisit again in this chapter, and elsewhere in the remainder of the book):

step 1. *Build your general model using PIMs, if you can. Recall that models with multiple groups based on PIMs assume interaction effects among levels of your groups. This is often – but not always – the basis for the general model in your candidate model set*

step 2. *Once you have the general model constructed using PIMs, then try to build the exact same model using the design matrix approach. Run this model. You'll know your design matrix is correct if the deviance for the model fit with the design matrix approach is the same as the deviance for the same model constructed using PIMs* (note*: the deviances should be the same – but the number of estimated parameters* ***MARK*** *reports might differ due to differences in the link function used)*.

step 3. *Once you've successfully built your general model with the design matrix, delete the same general model you built with PIMs – you don't want two copies of the same model in your results browser. Then, build all other models in the candidate model set by manipulating the design matrix of your general model.*

These 3 steps are good, general recommendations on where to start. Once you've built your general model using the design matrix, you can quickly and easily construct reduced parameter models – including models with additive effects – simply by manipulating the design matrix. This invariably means deleting or modifying one or more columns in the design matrix. Once you get the hang of this approach, it will become fairly automatic to you.

6.8. Constraining with 'real' covariates

In the previous sections, we've considered variation in one parameter or another over time – implicitly, we've been treating time as a 'classification' variable, and looking for heterogeneity among 'time intervals' in a particular parameter. Generally, though, we're not interested in whether or not there is variation over time, but whether this variation over time corresponds to one or more 'other variables' (covariates) which we think might cause (or contribute to) the variation we observe. In other words, our interest is typically in the causes of the temporal variation, not the variation itself.

We can address this hypothesis (i.e., that variation in some parameter over time reflects variation in some covariate) by building a linear model where the parameter estimates are constrained to be linear functions of one or more covariates. This is the subject of this section – constraining parameters to be functions of 'real' variables (in the mathematical sense of real), as opposed to simple 'dummy' or other integer variables. For example, suppose we have measured some other variable, such as total precipitation, or measures of annual food abundance, which can take on 'real' or 'fractional' values. Clearly, we might want to test the hypothesis that a model where one or both parameters are constrained to be linear functions of this type of covariate might be extremely useful. Fortunately, we have to learn nothing new in order to do this in **MARK** – all we need to do is put our 'real' covariates into our design matrix.

Consider the following example. Suppose you believe that capture rate is a function of the number of hours spent by observers in the field. This makes good intuitive sense – the more hours spent in the field, the more likely you might be to see a marked individual given that it is still alive. So, one way we might increase the precision of our estimate of recapture probability is to constrain the recapture parameters

to be linear functions of number of observations hours at each occasion. Recall that parsimonious modeling of recapture probability will also influence our estimates of survival probability as well.

These data are unavailable for the Dipper data set, so we'll 'make up' some values, just for purposes of illustrating this. Here are our 'data':

```
Occasion  2     3     4     5     6     7
hours     12.1  6.03  9.1   14.7  18.02 12.12
```

Now, we simply need to construct the correct design matrix. One slight twist here is that for the first time we're going to apply a constraint to the recapture probabilities, rather than the survival probabilities. This is no more difficult than what we've already done – all you need to do is identify the index values of the parameters you want to constrain. Recall that for the Dipper data set, with 2 sexes and 7 occasions, parameters $1 \rightarrow 6$ are male survival, parameters $7 \rightarrow 12$ are female survival, parameters $13 \rightarrow 18$ are male recapture probability, and finally, parameters $19 \rightarrow 24$ are female recapture probability. Thus, if we want to constrain our recapture probabilities to be linear functions of observer hours, we're going to constrain parameters $13 \rightarrow 24$. This means that we're working in the lower-right quadrant of the design matrix.

Next, we need to decide on the model we want to test. Let's test the model where we allow the sexes to potentially differ, with full interaction. In other words,

$$\text{logit}(p) = \beta_1 + \beta_2(\texttt{SEX}) + \beta_3(\texttt{HOURS}) + \beta_4(\texttt{SEX.HOURS})$$

As you can see, it is exactly the same qualitative model 'structure' as in our earlier 'flood' example, with a different 'covariate' (`HOURS` instead of `FLOOD`). Given this similarity, you might guess the design matrix should look similar as well. In fact, as shown at the top of the next page, it is virtually an inverted mirror image of the design matrix you used for the flood analysis – now the upper-left quadrant has the time-specific coding we've seen before, and the lower-right quadrant (pictured) has the dummy variable coding for `INTCPT`, `SEX`, `HOURS`, and the `SEX.HOURS` interaction. The only real difference (other than being in the lower-right quadrant) is that instead of '`0`' or '`1`' to represent `FLOOD` states, we replace that column of '`0`' and '`1`' values with the 'real' number of observer hours. And, since these are simply different levels of a single factor (`HOURS`), we need only one column to code for `HOURS`.

13:p	1	1	12.1	12.1
14:p	1	1	6.03	6.03
15:p	1	1	9.1	9.1
16:p	1	1	14.7	14.7
17:p	1	1	18.02	18.02
18:p	1	1	12.12	12.12
19:p	1	0	12.1	0
20:p	1	0	6.03	0
21:p	1	0	9.1	0
22:p	1	0	14.7	0
23:p	1	0	18.02	0
24:p	1	0	12.12	0

However, as you'll remember from our earlier examples, if you change either of the 2 columns contained in the interaction term, you also need to change the values in the interaction term itself.

The recapture portion of the design matrix (i.e., the lower-right quadrant) is shown above. Note we still have 12 rows, and 4 columns for the recapture parameter (reflecting the number of variables in the constraint). Once you have the design matrix constructed, you proceed in precisely the same fashion as you did with the '`flood`' example we just covered – the only difference is that, in this example, you're concentrating on recapture probabilities, rather than survival.

begin sidebar

linear covariates and nested models: LRT revisited

You may recall that in Chapter 4, we introduced two different approaches to model selection – one based on classical 'hypothesis testing' (e.g., the likelihood ratio test – LRT), and the other on information theoretic approaches (AIC, BIC). Recall that the classical LRT requires that models be nested. What constitutes 'nested' in the case of models with one or more linear covariates? What are the options if models are not strictly nested?

Recall that in Chapter 4, we defined nested models as follows:

> **nested models**: *Two models are nested if one model can be reduced to the other model by imposing a set of linear restrictions on the vector of parameters.*

For example, consider models f and g, which we'll assume have the same functional form and error structure. For convenience, we'll express the data as deviations from their means (doing so eliminates the intercept from the linear model, since it would be estimated to be 0). These two models differ then only in terms of their regressors. In the following

$$f: \; Y = \beta_1 x_1 + \epsilon_0$$
$$g: \; Y = \beta_1 x_1 + \beta_2 x_2 + \epsilon_1$$

the model f is nested within model g because by imposing the linear restriction that $\beta_2 = 0$, model g becomes model f.

What about non-nested models? Again, in Chapter 4 we defined non-nested models as

> **non-nested models**: *Two models are non-nested, either partially or strictly* (discussed below)*, if one model cannot be reduced to the other model by imposing a set of linear restrictions on the vector of parameters*

Consider the following two models:

$$f: \; Y = \beta_1 x_1 + \beta_2 x_2 + \epsilon_0$$
$$g: \; Y = \beta_2 x_2 + \beta_3 x_3 + \epsilon_1$$

Models f and g are non-nested because even if we impose the restriction on model g that $\beta_3 = 0$, model g does not become model f.

In fact, in this example, models f and g are *partially non-nested*, because they have one variable in common (x_2). If the two models didn't share x_2, then they would be *strictly non-nested*.

However, you need to be somewhat careful in defining models as strictly non-nested. There are, in fact, two cases where models with different sets of regressors may not be strictly non-nested.

Consider the following two models:

$$f: Y = \beta_1 x_1 + \epsilon_0$$
$$g: Y = \beta_2 x_2 + \epsilon_1$$

If either β_1 or β_2 equals zero, then the models are nested. This is trivially true. Less obvious, perhaps, is the situation where one of more of the explanatory variables in one model may be written as a linear combination of the explanatory variables in the second model. For example, given the two models

$$f: Y = \beta_1 x_1 + \epsilon_0$$
$$g: Y = \beta_2 x_2 + \epsilon_1$$

consider a third model h where

$$h: Y = \beta_3 x_3 + \epsilon_2 = \beta_1 x_1 + \beta_2 x_2 + \epsilon_2$$

Then, perform the following hypothesis tests: model h versus model f (i.e., $\beta_2 = 0$ versus $\beta_2 \neq 0$), and model h versus model g (i.e., $\beta_1 = 0$ versus $\beta_1 \neq 0$).

OK, what about the situation we're considering here – linear models with one or more linear covariates? Consider the following linear model for some parameter θ corresponding to a 5 occasion study:

$$\theta = \beta_0 + \beta_1(\texttt{interval}_1) + \beta_2(\texttt{interval}_2) + \beta_3(\texttt{interval}_3)$$

Remember – 5 occasions = 4 intervals = (4−1) = 3 columns of dummy variables coding for the intervals ($\beta_1 \rightarrow \beta_3$).

Here is the design matrix (DM) corresponding to this time-dependent linear model:

intcpt	β_1	β_2	β_3
1	1	0	0
1	0	1	0
1	0	0	1
1	0	0	0

Now, suppose we wanted to constrain this model such that the estimate of the parameter in the first and third intervals was equal. How would we modify the DM to achieve this constraint? The key is remembering what each β_i column represents: β_1 represents the first interval (between occasion 1 and 2), β_2 represents the second interval (between occasion 2 and 3), and so on. So, to constrain the estimates for $\hat{\theta}$ to be the same for the first and third intervals (i.e., $\theta_1 = \theta_3$), we have to (i) eliminate one of the two columns corresponding to these intervals (either the β_1 or β_3 columns), and (ii) add a '1' dummy variable in the appropriate row to the remaining column. For example, in the following DM

intcpt	β_1	β_2
1	1	0
1	0	1
1	1	0
1	0	0

we have eliminated the β_3 column from the original DM, and added a dummy '1' in the 3rd row of column β_1 – recall that row 3 corresponds to interval 3. The presence of a '1' in the first and third rows in the β_1 column is what constrains $\theta_1 = \theta_3$. This is essentially the same sort of thing we did for the flood example we considered earlier in this chapter. We have constrained our time-dependent model in a particular way – using the linear constraint $\theta_1 = \theta_3$.

Similarly, what if we want to constrain $\hat{\theta}$ to be a linear function of some continuous covariate (say, rainfall). Our DM might now look like

intcpt	β_1
1	2.3
1	4
1	1.2
1	5

Here, we've constrained the estimates for each interval to be a linear function of the rainfall covariate – one β column. So, based on the criterion for 'nestedness' – where two models are nested if one model can be reduced to the other model by imposing a set of linear restrictions on the vector of parameters – these two constrained models we've just constructed are both nested within the more general time-dependent model. And, as such, these models could both be compared to the time-dependent model using an LRT.

end sidebar

6.8.1. A special case of 'real covariates' - linear trend

Although we are not usually interested in a simple demonstration of temporal variation in our parameter estimates (as discussed in the preceding section), there is one 'special' case where we might be. If our estimates are believed to be increasing or decreasing over time (i.e., showing a trend). We will now explore how to use **MARK** to test for 'trend' in the data – linear increase or decrease in survival or recapture. We created a simple data set (`LINEAR.INP`), which contains simulated data which we will use for our analysis of linear trend. There are 8 occasions in the data set. Assume that the 'simulated animals' are all marked as adults. We started with fitting the basic CJS, time-dependent model. Since the data were simulated, we can safely assume that this is an acceptable model, and fits the data (i.e., you can leave $\hat{c}$ at the default value of 1.000). Since there is only one group, this analysis is very easy, and should take you only a few minutes with **MARK**. At this stage, we'll assume you know the steps for fitting this model, so we'll proceed directly to the results. The deviance of the CJS model for these data was 185.388, and the AIC_c value was 7980.78.

The estimates for both survival and recapture probabilities are tabulated below:

survival		**recapture**	
Parm	*Estimate*	*Parm*	*Estimate*
1	0.7237	8	0.5492
2	0.7022	9	0.5040
3	0.6280	10	0.5403
4	0.5836	11	0.5256
5	0.6267	12	0.4316
6	0.4586	13	0.5428
7	0.5027	14	0.5027

Note that the value of the last estimate for both survival and recapture is the same (0.5027) – as you might have remembered, this is the β_8 term. Thus, for this model, we have 13 total potentially identifiable parameters. The deviance for the model was 185.39, and the AIC_c for this model is therefore 7980.78. Note that the time-specific estimates show a clear trend (not surprising, since they were simulated this way).

Now, let's proceed to see how to use **MARK** to fit a model with a linear trend – this will allow us to formally test our hypothesis that there may be a decline in survival over time. Doing this in **MARK** is very straightforward. Mechanically, we again make some simple modifications to the CJS design matrix. First, what are the index values of the survival parameters we want to constrain (i.e., constrain to be a linear function of time)? Clearly, parameters $1 \rightarrow 7$. Now, as hinted in the last section, to fit a linear trend model, we need to modify the design matrix – how would we do this? Think back to the first example using the Dipper data set – the `FLOOD` analysis. Recall that in the design matrix, we had 4 columns of numbers in that file: one for the intercept, one for `SEX`, one for `FLOOD`, and one for the interaction term (`SEX.FLOOD`). Concentrate on the `FLOOD` column. We coded `FLOOD` as a simple binary variable: there was either a flood ('`1`') or there wasn't ('`0`'). In our present example, however, things aren't quite so simple. We are trying to build a model with a linear trend. In other words, a systematic change in survival (up or down) through time.

What do we know about a 'trend', and how does it differ from the flood example? By definition, a trend has a slope which is significantly different from 0. Given that the slope differs from 0, then on average, $y_{(i-1)} < y_i < y_{(i+1)}$ for an increasing trend with (i), and the reverse for a decreasing trend. Second, a trend (if linear) is 'continuous' through time – it is not a simple binary condition, as was the case with flood. Thus, we need to code a 'trend' through time in such a way that it meets these 2 conditions. As it turns out, it is very simple to do this, although you may have to take a few minutes to grasp the logical connection. To code for a linear trend, all you need to do is write a series of ordinal, evenly spaced increasing (or decreasing) numbers, 1 through n (where n is the number of occasions you want to fit the 'trend' to). You don't have to start with the number 1, but you do need to use the sequence {starting value} + 1, {starting value} + 2, and so on. So, what would the survival elements of the design matrix look like for this 8 occasion study?

Just like this:

1	1
1	2
1	3
1	4
1	5
1	6
1	7

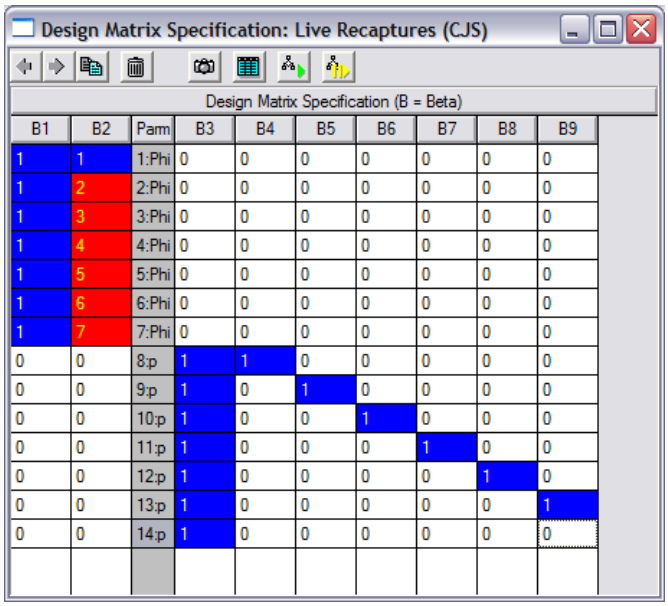
Hmmm. . .pretty strange looking (perhaps) – what do we have here? The first column corresponds to the intercept, while the second column is the dummy variable coding for the linear trend. In other words,

$$\text{logit}(\varphi) = \texttt{INTERCEPT} + \beta_1(\texttt{T})$$

where `T` (commonly referred to as 'cap T') indicates a linear trend (we use a capital `T` for trend, to distinguish a trend model from a simple time-dependent model, which is usually indicated using a lower-case `t`).

So, only 2 β terms: one for the intercept, and one for the linear relationship between the response

variable (φ, in this case), and the value for 'trend'. You should see the connection (at least structurally) between this linear model, and how we coded for the 'effort' covariate in the previous section. The order of the numbers (1 to 7, or 7 to 1) makes no difference – **MARK** will simply use the numbers to fit a linear trend – it will let the data determine if the trend is up or down (if any trend exists). Get it? The design matrix for this model ('`Phi(linear)p(t)`') is shown below:



B1	B2	Parm	B3	B4	B5	B6	B7	B8	B9
1	1	1:Phi	0	0	0	0	0	0	0
1	2	2:Phi	0	0	0	0	0	0	0
1	3	3:Phi	0	0	0	0	0	0	0
1	4	4:Phi	0	0	0	0	0	0	0
1	5	5:Phi	0	0	0	0	0	0	0
1	6	6:Phi	0	0	0	0	0	0	0
1	7	7:Phi	0	0	0	0	0	0	0
0	0	8:p	1	1	0	0	0	0	0
0	0	9:p	1	0	1	0	0	0	0
0	0	10:p	1	0	0	1	0	0	0
0	0	11:p	1	0	0	0	1	0	0
0	0	12:p	1	0	0	0	0	1	0
0	0	13:p	1	0	0	0	0	0	1
0	0	14:p	1	0	0	0	0	0	0

Note that this model has 9 parameters (1 for the intercept, 1 for the slope of the 'regression' of survival on time as a linear covariate, and 7 recapture parameters – no non-identifiable product β terms. Make sure you know why!). The deviance was 189.53, and the AIC_c was 7976.88. The model where survival was constrained to vary linearly with time (note we do not specify increase or decrease) is a more parsimonious model than the initial time-dependent model – in fact, it is approximately 7 times better supported by the data. This is perhaps not surprising – the data were simulated with a decline in survival!

Now, one subtle variation in this theme: suppose that we want to test for a non-linear trend. How would we do this? Well, there are several ways to analyze non-linear relationships. Perhaps the easiest is to use multiple regression, fitting a series of power terms to the function. For example, a comparison of the model $Y = X + X^2$ to model $Y = X$ is a formal test of the significance of the X^2 term. If the X^2 term is not significant, and if the model $Y = X$ fits the data, then we can conclude that the relationship is linear. How would we do this? In fact, it is very simple. All you need to do is add another column to your design matrix file to accommodate the X^2 term. It's that easy!

Warning: While the mechanics of what we've just demonstrated for fitting a 'trend' model appear straightforward, there is a conceptual limitation to this particular approach which you need to consider. Fitting a linear trend model using the approach we described in this section 'forces' the parameter estimates to fall precisely on a line. Clearly, this 'statistical constraint' enforces a 'biological' hypothesis which is implausible – the estimates are no more likely to fall precisely on a line than they are to be exactly the same in a 'constant over time' model. In addition, inference concerning the estimated 'slope' of the trend from a 'cap T' model is problematic – in such models, the estimated standard errors are based only on sampling variation, and would be biased low compared to a direct regression on

true estimates (which is clearly not possible because the true estimates are not known). However, an approach based on *random effects* solves this problem. In this context, random effects models assume that 'true' annual estimates do not fall exactly on any simple, smooth model; the deviation of estimates of some parameter from such models are treated as random. So, rather than assume the estimates fall precisely on a straight line (i.e., applying a simple 'cap T' model approach), the random effects regression assumes that the actual 'true' estimates vary randomly around some trend line (where the trend line represents the mean estimate if multiple samples were available from the same range of years in the data). Random effects models in **MARK** are covered in detail in Appendix D.

begin sidebar

Another type of 'trend': the cumulative logit link

The cumulative logit link (CLogit) function is useful for constraining a set of parameters to monotonically increase. Suppose that you desire the relationship of $S_1 \leq S_2 \leq S_3$, but do not want to enforce the relationship on the logit scale that

$$\text{logit}[S_2] - \text{logit}[S_1] = \text{logit}[S_3] - \text{logit}[S_2]$$

as a trend model (discussed in the preceding section) would do.

The CLogit link would generate this relationship as:

$$S_1 = \frac{e^{\beta_1}}{1 + e^{\beta_1} + e^{\beta_2} + e^{\beta_3}} \qquad S_2 = \frac{e^{\beta_1} + e^{\beta_2}}{1 + e^{\beta_1} + e^{\beta_2} + e^{\beta_3}} \qquad S_3 = \frac{e^{\beta_1} + e^{\beta_2} + e^{\beta_3}}{1 + e^{\beta_1} + e^{\beta_2} + e^{\beta_3}}$$

To use the CLogit link, you have to specify a separate CLogit link for each set of parameters that are to be constrained. In addition, you also have to specify the order of the parameters for the set.

For the above example, the link function for each of the 3 survival probabilities would be:

```
S(1): CLogit(1,1)
S(2): CLogit(1,2)
S(3): CLogit(1,3)
```

If you have a second set of parameters (in the same model) that you also want to enforce a monotonic *increase* on, say $S_4 \leq S_5 \leq S_6$, the appropriate links would be:

```
S(4): CLogit(2,1)
S(5): CLogit(2,2)
S(6): CLogit(2,3)
```

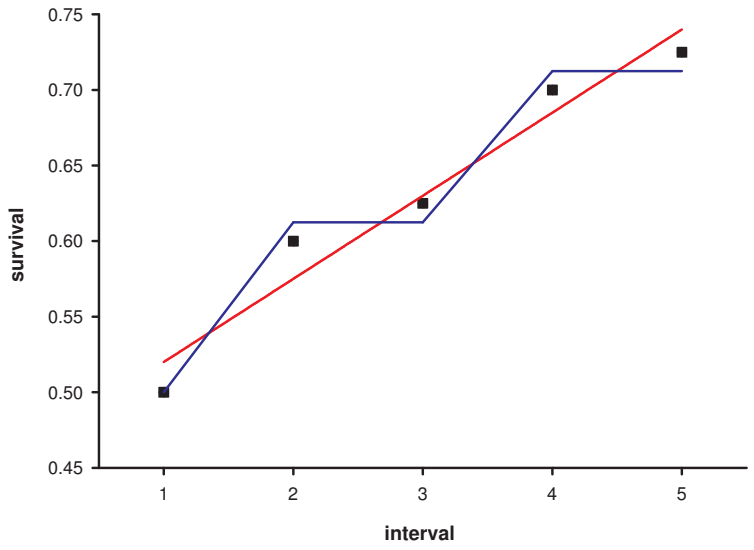
Note that you can force a monotonic *decrease* by reversing the order of the real parameters in the CLogit set. For example, to enforce a monotonic decrease on parameters S_1, S_2, and S_3, you would use

```
S(1): CLogit(1,3)
S(2): CLogit(1,2)
S(3): CLogit(1,1)
```

You specify these link functions by selecting the '`Parm-Specific`' choice from the '`Run Window`' list of link functions, and then entering the appropriate specification in the edit box next to the parameter name.

We'll demonstrate the use of the CLogit link by means of a worked example, based on simulated live encounter data (6 occasions, 250 individuals marked and released at each occasion), where apparent

survival φ tends to increase monotonically, while the encounter probability p is constant over time (the simulated data are contained in `clogit_demo.inp`). The true values for the survival parameters are: $S_1 = 0.500$, $S_2 = 0.600$, $S_3 = 0.625$, $S_4 = 0.700$ and $S_5 = 0.725$. Note that $S_2 \cong S_3$, and $S_4 \cong S_5$. Thus, even though there is an obvious tendency for survival to increase over time, the increase is clearly not strictly linear, as indicated by the square symbols in the following figure:



However, the question is whether a model which constrains the estimates to be strictly linear (red line) is a better fit to the data than one which allows for possible equality between some of the estimates (blue line).

Recall that fitting a linear trend using the design matrix forces the reconstituted estimates to fall on a perfectly straight line. So, in this example, it might seem possible (perhaps likely) that a model based on the cumulative logit link (which allows for possible equality between some of the estimates) may prove to be more parsimonious than a strictly linear trend model (even though the latter will often have fewer parameters).

First, build model $\{\varphi_t p_.\}$, and add the results to the browser. Note that the real parameter estimates from this model (shown below)

mrk9670z.tmp - Notepad

File Edit Format View Help

CLogit demo

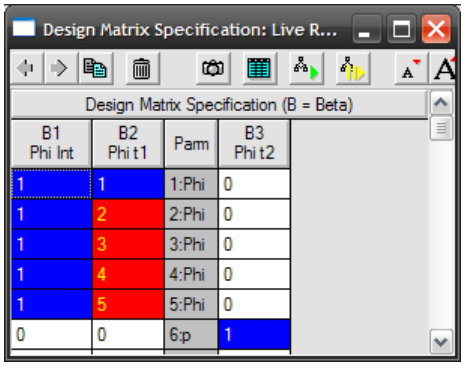
Real Function Parameters of {phi(t)p(.)}

Parameter	Estimate	Standard Error	95% Confidence Interval Lower	Upper
1:Phi	0.5354208	0.0429860	0.4509609	0.6178968
2:Phi	0.6972772	0.0399885	0.6137379	0.7695335
3:Phi	0.5971751	0.0328316	0.5315093	0.6595326
4:Phi	0.7669756	0.0375781	0.6855091	0.8324955
5:Phi	0.7378645	0.0449388	0.6409564	0.8161203
6:p	0.5011069	0.0201832	0.4616233	0.5405768

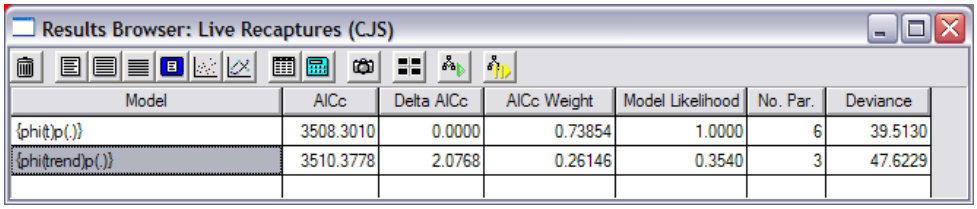
are not particularly close to the true parameter values used to simulate the data. Given the relatively

small sample size of newly marked individuals released at each occasion, this is perhaps not surprising.

Now, let's fit two additional models: model $\{\varphi_{\text{trend}} p.\}$ (simple linear trend on apparent survival), and model $\{\varphi_{\text{CLogit}} p.\}$, where we use the cumulative logit link to impose an increasing, ordinal – but not strictly linear – structure on the apparent survival values. If you followed the material covered in this section, fitting the simple linear trend model should be easy. Here is the design matrix corresponding to the trend model:



Go ahead, fit this model, and add the results to the browser:



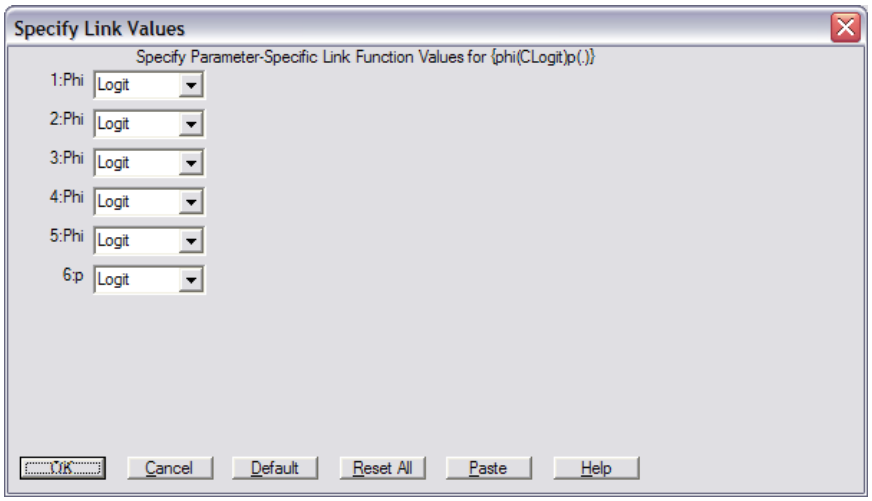
We see clearly that model $\{\varphi_{\text{trend}} p.\}$ is not particularly well-supported by the data (at least, relative to model $\{\varphi_t p.\}$).

Now, let's fit model $\{\varphi_{\text{CLogit}} p.\}$, where we impose an increasing, ordinal constraint on the estimates of apparent survival. In other words, we're constraining the estimates of $S_1, S_2, \cdots, S_5$ such that

$$S_1 \leq S_2 \leq S_3 \leq S_4 \leq S_5$$

Now, if you look closely at the above, you'll see that there are a very large number of possible models which would satisfy this constraint – **MARK** will effectively test all of them, and select the most parsimonious of the set.

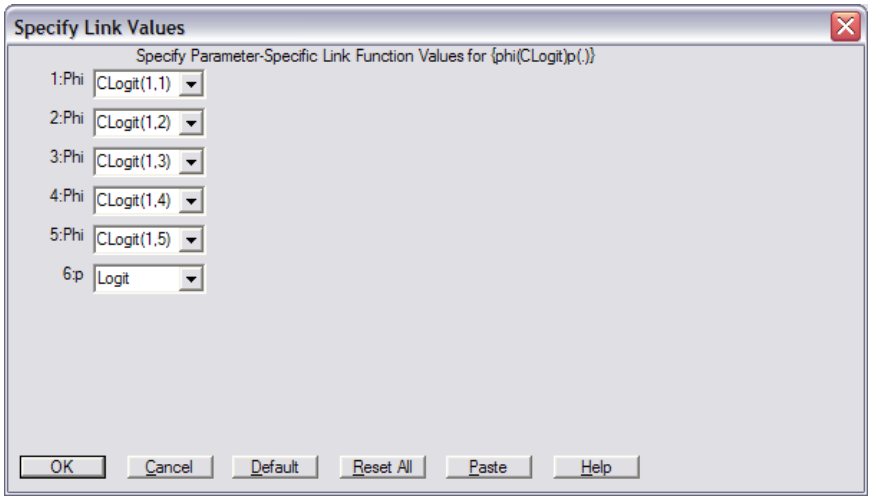
To build this model in **MARK**, first retrieve model `phi(t)p(.)` from the browser. You can do this easily by selecting the model in the browser, right-clicking, and selecting '**Retrieve**'. Then, select '**Run**' again, and change the name of the model to `phi(CLogit)p(.)`. Now, before clicking the '**OK to Run**' button, we need to specify the cumulative logit link for the 5 apparent survival parameters. To do this, you need to select the '**Parm-specific**' radio button option in the list of link functions. Now, when you click the '**OK to Run**' button, **MARK** will respond with a window where you will specify the parameter specific link functions you want to use (shown below). Note that in in this case, **MARK** defaults to the logit link for all parameters (the default link for each parameter will either be the sin or logit link, depending on whether or not you are using an identity design matrix).



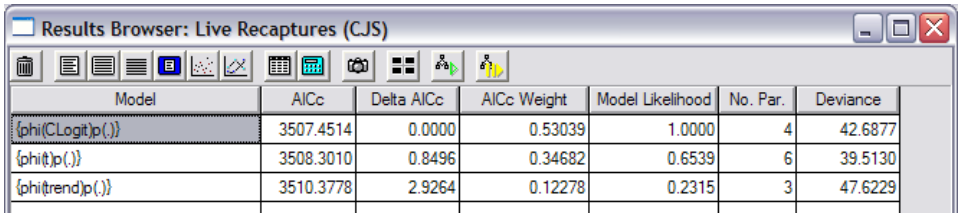
To fit our model, we need to change from the default logit link to the cumulative logit link, for parameters $1 \rightarrow 6$, corresponding to $\varphi_1 \rightarrow \varphi_5$. To enforce a monotonic increase in apparent survival, subject to the condition that

$$S_1 \leq S_2 \leq S_3 \leq S_4 \leq S_5$$

we simply specify the CLogit link for each of the survival parameters. For this example, the completed link specification window is shown at the top of the next page. Note that you *cannot* select the CLogit link function from the drop-down list of link functions like as with all of the other link functions, because you have to specify the set and the order of the parameter within the set. Therefore, you have to *manually* enter the link function in each edit box next to the real parameter value to which it pertains. This is the most logical method to provide the user the flexibility needed to select the parameters for each CLogit set, and still specify the order of the increase of the real parameters.

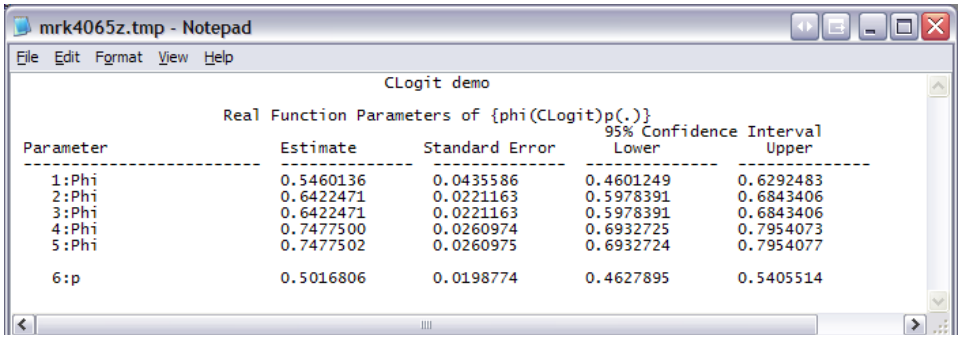


Once you've specified the link functions, go ahead and run the model, and add the results to the browser:



Model	AICc	Delta AICc	AICc Weight	Model Likelihood	No. Par.	Deviance
{phi(CLogit)p(.)}	3507.4514	0.0000	0.53039	1.0000	4	42.6877
{phi(t)p(.)}	3508.3010	0.8496	0.34682	0.6539	6	39.5130
{phi(trend)p(.)}	3510.3778	2.9264	0.12278	0.2315	3	47.6229

We see clearly that the model using the cumulative logit link has considerable AIC weight in the data, relative to the other models in the model set. The estimates from this model are shown below:



You see that the results follow the basic constraint specification

$$S_1 \leq S_2 \leq S_3 \leq S_4 \leq S_5$$

In this case, the most parsimonious model was one where

$$S_1 \leq S_2 = S_3 \leq S_4 = S_5$$

which seems quite reasonable given that for the true parameter values (noted a few pages back), $S_2 \cong S_3$, and $S_4 \cong S_5$.

So, how many parameters are estimated, subject to the constraint that

$$S_1 \leq S_2 \leq S_3 \leq S_4 \leq S_5$$

Given that the most parsimonious ML estimates for these simulated data subject to this constraint are

$$S_1 \leq S_2 = S_3 \leq S_4 = S_5$$

it is clear that 3 parameters are estimated.

Now, it is worth noting here that what we have done is essentially equivalent to a 'all subsets' regression – **MARK** has simply found the most parsimonious model from the set of models specified by the constraint that

$$S_1 \leq S_2 \leq S_3 \leq S_4 \leq S_5$$

This may be potentially usefully for finding a parsimonious model for improving precision of reconstituted parameter estimates, or if you believe that there is a general monotonic increase or decrease

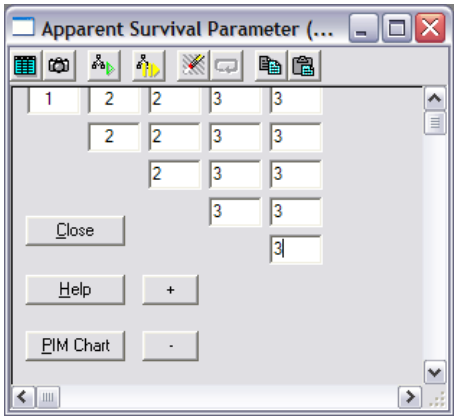
in some parameter, where you have no *a priori* expectation to the particular form of the function (other than it being monotonic in one direction or the other). In this case, the most parsimonious model was one where

$$S_1 \leq S_2 = S_3 \leq S_4 = S_5$$

was *not* motivated by a particular *a priori* hypothesis.

If in fact you had reason to include a model with this constraint in your model set (i.e., if you had an *a priori* expectation that $S_2 = S_3$ and $S_4 = S_5$, with a monotonic increase from $S_1 \rightarrow S_2 = S_3 \rightarrow S_4 = S_5$), then it would be more appropriate to (i) first construct a PIM with the basic structure $\{S_1, S_2 = S_3, S_4 = S_5\}$ (shown below) to which you then (ii) apply a cumulative logit constraint to parameters $1 \rightarrow 3$. If you try it for this example problem, you'll see that you generate *exactly* the same reconstituted parameter estimates as you did from the cumulative logit link applied to the fully time-dependent PIM.

However, by applying the cumulative logit to the PIM that reflects our *a priori* beliefs about equality of certain parameters, then we avoid the risk of having to concoct a *post hoc* 'story' (not withstanding their frequent entertainment value) to explain the particular CLogit model that **MARK** has found to be most parsimonious. The distinction here is subtle, but important.



end sidebar

6.9. More than 2 levels of a group

It is not uncommon to have more than 2 levels of a classification variable in an analysis. For example, you may have a control and 2 or more treatment groups. How would you code the design matrix for such a situation? In fact, it's easy (well, relatively), if you remember some basic principles from analysis of variance (ANOVA). The number of columns used to characterize group differences (e.g., belonging to one group or not) will always equal the number of dummy variables (coded '0' or '1') that you need to characterize group differences. For any variable that we treat as 'categorical' or 'classification' (i.e., both COLONY and TIME in the swift example) with k levels, the number of columns needed is $(k-1)$, which happens to be the numbers of degrees of freedom associated with a factor in a standard ANOVA (note – it's *not* a coincidence). In short, the number of columns (hence, the number of dummy variables) needed equals the degrees of freedom associated with that variable. So, the minimum number of columns of dummy variables needed to specify our model are defined by the number of degrees of freedom for each factor, plus any interaction terms.

Of course, it **is** possible to specify a model with more columns than the minimum set we've just described. Would this be wrong? Not exactly – your estimates would be 'correct', but it becomes very difficult to count (separately) identifiable parameters. And since counting parameters is essential to model testing, using more columns than necessary in your design matrix to specify the model should be avoided.

Consider an example of a study over 5 occasions, where we have 3 'groups', or levels of our 'main effect'. Thus, we need (3-1) = 2 columns of 'dummy variables' to specify group association. Again, it is important to understand the logic here: we need $(n - 1) = (3 - 1) = 2$ columns, because we have an intercept in the model – the intercept codes for one level of the treatment, and the other two columns code for the remaining two levels of the treatment.

Suppose we also have a quantitative covariate (say, hours of observation). Linear terms have 1 degree of freedom, so one column of covariate values. Finally, for the interaction, we need $(3 - 1) \times (1) = 2$ columns.

Here is the design matrix – we have formatted it slightly to emphasize the 'logical connection' amongst the columns.

INTCPT	GROUP		HOURS	GROUP.HOURS	
1	0	0	1.1	0	0
1	0	0	0.2	0	0
1	0	0	3.4	0	0
1	0	0	4.1	0	0
1	0	1	1.1	0	1.1
1	0	1	0.2	0	0.2
1	0	1	3.4	0	3.4
1	0	1	4.1	0	4.1
1	1	0	1.1	1.1	0
1	1	0	0.2	0.2	0
1	1	0	3.4	3.4	0
1	1	0	4.1	4.1	0

The first column is the intercept. The next 2 columns on the left indicate group: group is identified depending upon the pair of dummy variables across these 2 columns: '0 0', '0 1', and '1 0'. The middle column (of the 5 total columns), is the 'covariate' column. The last 2 columns are the interaction columns (interaction of group and covariate). Remember, the interaction term can be thought of as a 'multiplication term' – the product of the various factors contained in the interaction. Since this interaction is the interaction of 'group' (2 columns) and 'covariate' (1 column), then we have $(2 \times 1) = 2$ columns for the interaction. We simply multiply each element of the group column vectors by its corresponding element in the 'hours' column vector. Therefore, 6 columns total.

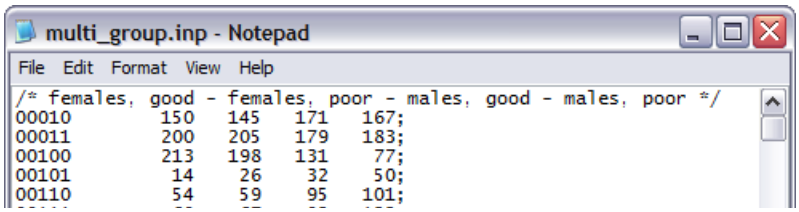
Now, if we were applying this design matrix, and we wanted to test for the significance of the interaction term, we would first run the constraint using all 6 columns in the matrix, and then a second time using only the first 4 columns – 4 because we want to drop the interaction term, which is 'stored' in columns 5 and 6.

Note that it is important to use the minimum number of columns of '0' or '1' dummy variables to characterize your groups. Why? Because each column in your matrix becomes a slope, which is an estimated parameter. Our goal is to use as few parameters as possible to specify a model. Extra columns wouldn't make your model 'wrong', but would make the counting of (separately) identifiable parameters more difficult.

It is also important to note that all that is necessary is that you use $n-1$ columns to code the 'group' variable (in this case, $(3-1) = 2$ columns). However, the actual 'dummy variable' coding you use to specify group is arbitrary. For example, in the preceding example, we used '0 0' for group 1, '0 1' for group 2, and '1 0' for group 3. However, we could have used '1 1' for group one, '1 0' for group 2, and '0 1' for group 3, or any of a number of other combinations. They would all yield the same results (although, clearly, the coding in the interaction columns will change from the preceding example to reflect whatever coding you use for group columns). Try a few examples to confirm this for yourself.

6.10. > 1 classification variables: *n*-way ANOVA

Back in Chapter 2, we briefly considered the formatting of the INP file for situations where you have > 1 classification factors. We considered an example where both males and females were sampled at each of two colonies: a good colony, and a poor colony. Thus, two classification factors: SEX and COLONY. Recall that in the input file, we included a frequency column for each (SEX.COLONY) combination: one frequency column for females from the good colony, one frequency column for females from the poor colony, one frequency column for males from the good colony, and finally, one frequency column for males from the poor colony. We simulated a simple data set (MULTI_GROUP.INP) including these 4 combinations. A portion of the INP file is shown below:



Here, we focus on building models which have both SEX and COLONY effects. Suppose, for example, we want to build the following linear model for φ:

$$\varphi = \text{SEX} + \text{COLONY} + \text{TIME} + \text{SEX.TIME} + \text{COLONY.TIME} + \text{SEX.COLONY} + \text{SEX.COLONY.TIME}$$

In other words, a 3-way ANOVA (in effect, TIME is the third classification variable in this analysis).

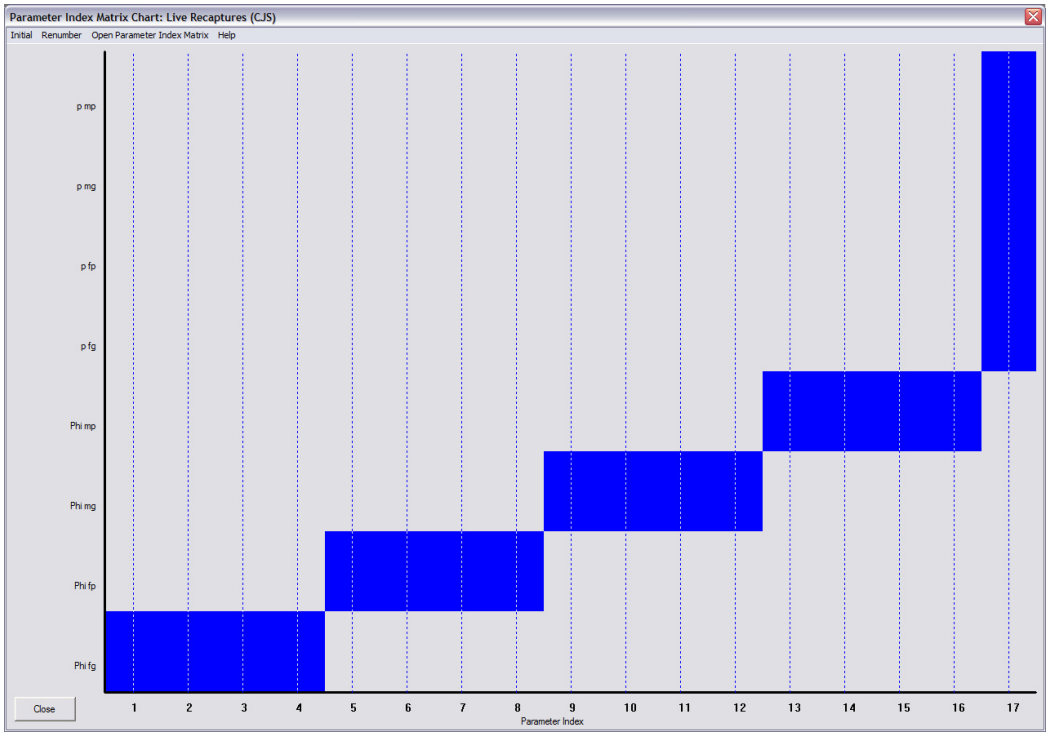
How do we go about building this model? Start program **MARK**, and begin a new project. Select MULTI_GROUP.INP. Specify 5 occasions. Now, for your first challenge – how many attribute groups? You might think either 2, or 3. Well, perhaps you're comfortable enough now with **MARK** to think just two – SEX and COLONY (realizing that TIME is an implicit attribute, and doesn't need to be counted).

Unfortunately, you'd not be correct. In fact, there are 4 attribute groups – corresponding to each of the 4 SEX.COLONY combinations (i.e., the number of frequency columns in the INP file). So, you need to tell **MARK** that there are 4 attribute groups.

Next, what to call them? In the INP file (above) we see that the first two frequency columns are for females sampled at the good and poor colonies, respectively, and the last two frequency columns are for males, sampled at the good and poor colonies, respectively. So, let's label the 4 attribute groups as FG, FP, MG, and MP, respectively.

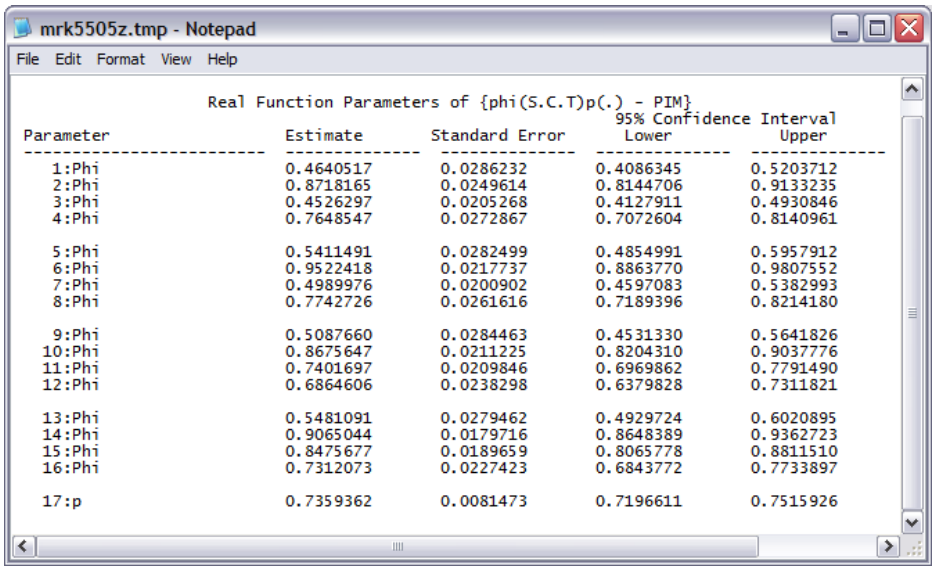
Now, let's build our model – to simplify, let's assume that the encounter probability p is the same for both sexes and both colonies, and constant over time.

The PIM chart reflecting our general model is shown below.



Study this PIM chart carefully – make sure you see the connection between the PIM chart, and the model we're trying to build. Go ahead and run the model – call it '`phi(S.C.T)p(.) - PIM`'. We add the `PIM` label to the model name to remind us that the model was built using the PIM chart.

Once the numerical estimation is complete, add the results to the browser. Here are the real estimates from this model (we've added some blank lines to make the groupings more obvious).



Real Function Parameters of {phi(S.C.T)p(.) - PIM}

Parameter	Estimate	Standard Error	95% Confidence Interval Lower	Upper
1:Phi	0.4640517	0.0286232	0.4086345	0.5203712
2:Phi	0.8718165	0.0249614	0.8144706	0.9133235
3:Phi	0.4526297	0.0205268	0.4127911	0.4930846
4:Phi	0.7648547	0.0272867	0.7072604	0.8140961
5:Phi	0.5411491	0.0282499	0.4854991	0.5957912
6:Phi	0.9522418	0.0217737	0.8863770	0.9807552
7:Phi	0.4989976	0.0200902	0.4597083	0.5382993
8:Phi	0.7742726	0.0261616	0.7189396	0.8214180
9:Phi	0.5087660	0.0284463	0.4531330	0.5641826
10:Phi	0.8675647	0.0211225	0.8204310	0.9037776
11:Phi	0.7401697	0.0209846	0.6969862	0.7791490
12:Phi	0.6864606	0.0238298	0.6379828	0.7311821
13:Phi	0.5481091	0.0279462	0.4929724	0.6020895
14:Phi	0.9065044	0.0179716	0.8648389	0.9362723
15:Phi	0.8475677	0.0189659	0.8065778	0.8811510
16:Phi	0.7312073	0.0227423	0.6843772	0.7733897
17:p	0.7359362	0.0081473	0.7196611	0.7515926

Now, we want to try to construct this model using the design matrix approach (since we want to be able to use the flexibility of the design matrix to build models we can't build with the PIMs).

First – how many columns should the design matrix have? Well, if you 'cheat' and look at the results browser, you might guess 17 – one column for each of the estimated parameters. But, we want to confirm our hunch – we do this by writing out the linear model in full. Remember, SEX has two levels (male and female), so 1 column for sex. Similarly, COLONY has two levels (good and poor), so again, 1 column for COLONY. There are 5 occasions, so 4 TIME intervals, and thus we need 3 columns to code for TIME. Finally, we need to code for the various interaction terms as well. Remember, there are interactions of SEX.TIME, COLONY.TIME, SEX.COLONY and SEX.COLONY.TIME.

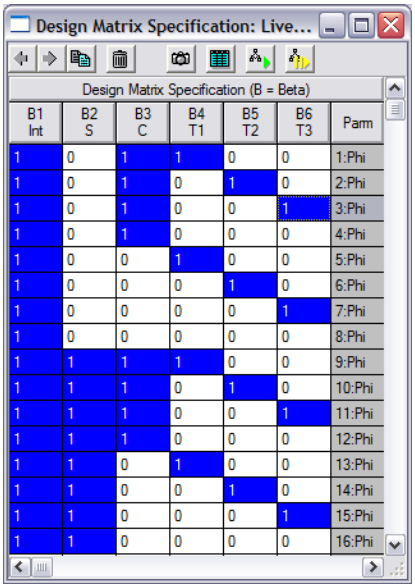
Here is the linear model:

$$\begin{aligned}\text{logit}(\varphi) = \beta_1 &+ \beta_2(\texttt{SEX}) + \beta_3(\texttt{COLONY}) \\ &+ \beta_4(\texttt{T}_1) + \beta_5(\texttt{T}_2) + \beta_6(\texttt{T}_3) \\ &+ \beta_7(\texttt{S.T}_1) + \beta_8(\texttt{S.T}_2) + \beta_9(\texttt{S.T}_3) \\ &+ \beta_{10}(\texttt{C.T}_1) + \beta_{11}(\texttt{C.T}_2) + \beta_{12}(\texttt{C.T}_3) \\ &+ \beta_{13}(\texttt{S.C}) \\ &+ \beta_{14}(\texttt{S.C.T}_1) + \beta_{15}(\texttt{S.C.T}_2) + \beta_{16}(\texttt{S.C.T}_3)\end{aligned}$$

So, we see that there are 16 β terms for φ, plus 1 β term for the constant encounter probability p, for a total of 17 columns (looks like our guess was correct). Now, let's build the design matrix.

We'll start by adding the INTCP, SEX, COLONY and TIME columns to the design matrix. As you no doubt by now realize, there is no 'hard rule' for how you code the various effects in the design matrix – as long as the coding, and the number of columns, are consistent with the model you're trying to fit, and the data in the INP file.

Here is one possible dummy variable coding for these terms:



B1 Int	B2 S	B3 C	B4 T1	B5 T2	B6 T3	Parm
1	0	1	1	0	0	1:Phi
1	0	1	0	1	0	2:Phi
1	0	1	0	0	1	3:Phi
1	0	1	0	0	0	4:Phi
1	0	0	1	0	0	5:Phi
1	0	0	0	1	0	6:Phi
1	0	0	0	0	1	7:Phi
1	0	0	0	0	0	8:Phi
1	1	1	1	0	0	9:Phi
1	1	1	0	1	0	10:Phi
1	1	1	0	0	1	11:Phi
1	1	1	0	0	0	12:Phi
1	1	0	1	0	0	13:Phi
1	1	0	0	1	0	14:Phi
1	1	0	0	0	1	15:Phi
1	1	0	0	0	0	16:Phi

Look closely. The INCTP is coded by a column of 16 '1's. There are 4 intervals, and 4 attribute groups, so 16 underlying φ parameters. Next, the SEX column (labeled S). We'll let '1' represent males, and '0'

represent females. Since the first 2 frequency columns in the INP file represent females, then the first 8 elements of the SEX column are 0's, followed by 8 1's for the males. Next, the COLONY column (labeled C). Now, remember that in the INP file, the frequency columns were 'good' and 'poor' colonies for the females, followed by 'good' and 'poor' colonies for the males. Here, we've used a '1' to indicate the 'good' colony, and a '0' to represent the 'poor' colony, alternating for each sex. Next, the 3 columns coding for TIME (labeled T1, T2 and T3), in the standard way (using reference cell coding) – here, we've set the final time interval (between occasion 4 and 5) as the reference interval.

What about interactions? Well, let's start with the easy ones: S.T, C.T, and S.C. Look closely at the figure at the top of the next page – for your convenience, we've labeled the columns in the design matrix so you can see which columns correspond to which interaction terms: columns 7 → 9 correspond to the SEX.TIME interactions, columns 10 → 12 correspond to the COLONY.TIME interactions, and column 13 corresponds to the SEX.COLONY interaction.

Design Matrix Specification (B = Beta)

B1 Int	B2 S	B3 C	B4 T1	B5 T2	B6 T3	B7 S.T1	B8 S.T2	B9 S.T3	B10 C.T1	B11 C.T2	B12 C.T3	B13 S.C	Parm
1	0	1	1	0	0	0	0	0	1	0	0	0	1:Phi
1	0	1	0	1	0	0	0	0	0	1	0	0	2:Phi
1	0	1	0	0	1	0	0	0	0	0	1	0	3:Phi
1	0	1	0	0	0	0	0	0	0	0	0	0	4:Phi
1	0	0	1	0	0	0	0	0	0	0	0	0	5:Phi
1	0	0	0	1	0	0	0	0	0	0	0	0	6:Phi
1	0	0	0	0	1	0	0	0	0	0	0	0	7:Phi
1	0	0	0	0	0	0	0	0	0	0	0	0	8:Phi
1	1	1	1	0	0	1	0	0	1	0	0	1	9:Phi
1	1	1	0	1	0	0	1	0	0	1	0	1	10:Phi
1	1	1	0	0	1	0	0	1	0	0	1	1	11:Phi
1	1	1	0	0	0	0	0	0	0	0	0	1	12:Phi
1	1	0	1	0	0	1	0	0	0	0	0	0	13:Phi
1	1	0	0	1	0	0	1	0	0	0	0	0	14:Phi
1	1	0	0	0	1	0	0	1	0	0	0	0	15:Phi
1	1	0	0	0	0	0	0	0	0	0	0	0	16:Phi

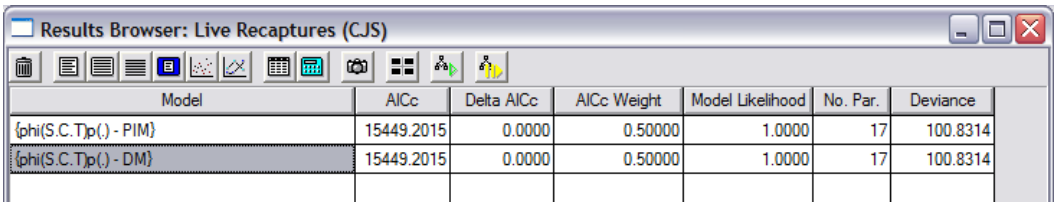
Finally, the S.C.T interaction, indicated in columns 14 → 16 in the following:

Design Matrix Specification (B = Beta)

B1 Int	B2 S	B3 C	B4 T1	B5 T2	B6 T3	B7 S.T1	B8 S.T2	B9 S.T3	B10 C.T1	B11 C.T2	B12 C.T3	B13 S.C	B14 S.C.T1	B15 S.C.T2	B16 S.C.T3	Parm	B17 p
1	0	1	1	0	0	0	0	0	1	0	0	0	0	0	0	1:Phi	0
1	0	1	0	1	0	0	0	0	0	1	0	0	0	0	0	2:Phi	0
1	0	1	0	0	1	0	0	0	0	0	1	0	0	0	0	3:Phi	0
1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	4:Phi	0
1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	5:Phi	0
1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	6:Phi	0
1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	7:Phi	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8:Phi	0
1	1	1	1	0	0	1	0	0	1	0	0	1	1	0	0	9:Phi	0
1	1	1	0	1	0	0	1	0	0	1	0	1	0	1	0	10:Phi	0
1	1	1	0	0	1	0	0	1	0	0	1	1	0	0	1	11:Phi	0
1	1	1	0	0	0	0	0	0	0	0	0	1	0	0	0	12:Phi	0
1	1	0	1	0	0	1	0	0	0	0	0	0	0	0	0	13:Phi	0
1	1	0	0	1	0	0	1	0	0	0	0	0	0	0	0	14:Phi	0
1	1	0	0	0	1	0	0	1	0	0	0	0	0	0	0	15:Phi	0
1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	16:Phi	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	17:p	1

The element in the lower right-hand corner of the completed design matrix codes for the constant encounter probability, p.

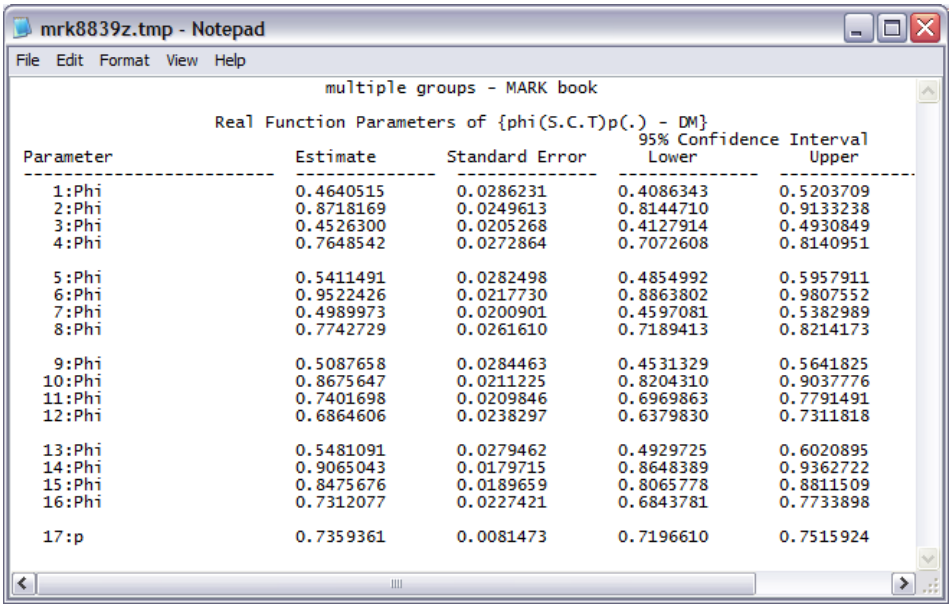
Now, go ahead and run this model, and label it '`phi(S.C.T)p(.) - DM`', with DM indicating it was constructed using a design matrix. Add the results to the browser:



Model	AICc	Delta AICc	AICc Weight	Model Likelihood	No. Par.	Deviance
{phi(S.C.T)p(.) - PIM}	15449.2015	0.0000	0.50000	1.0000	17	100.8314
{phi(S.C.T)p(.) - DM}	15449.2015	0.0000	0.50000	1.0000	17	100.8314

As expected, the two models yield identical results: the number of parameters and the model deviance is the same, regardless of whether or not the model was built using the PIMs, or via the design matrix.

As a final check, though, we want to look at the real estimates for our parameters from the model constructed using the following design matrix (shown below):




```
                        multiple groups - MARK book

                Real Function Parameters of {phi(S.C.T)p(.) - DM}
                                                          95% Confidence Interval
Parameter               Estimate       Standard Error      Lower           Upper
-------------------------  -------------  -------------  -------------  -------------
   1:Phi                   0.4640515      0.0286231      0.4086343      0.5203709
   2:Phi                   0.8718169      0.0249613      0.8144710      0.9133238
   3:Phi                   0.4526300      0.0205268      0.4127914      0.4930849
   4:Phi                   0.7648542      0.0272864      0.7072608      0.8140951

   5:Phi                   0.5411491      0.0282498      0.4854992      0.5957911
   6:Phi                   0.9522426      0.0217730      0.8863802      0.9807552
   7:Phi                   0.4989973      0.0200901      0.4597081      0.5382989
   8:Phi                   0.7742729      0.0261610      0.7189413      0.8214173

   9:Phi                   0.5087658      0.0284463      0.4531329      0.5641825
  10:Phi                   0.8675647      0.0211225      0.8204310      0.9037776
  11:Phi                   0.7401698      0.0209846      0.6969863      0.7791491
  12:Phi                   0.6864606      0.0238297      0.6379830      0.7311818

  13:Phi                   0.5481091      0.0279462      0.4929725      0.6020895
  14:Phi                   0.9065043      0.0179715      0.8648389      0.9362722
  15:Phi                   0.8475676      0.0189659      0.8065778      0.8811509
  16:Phi                   0.7312077      0.0227421      0.6843781      0.7733898

  17:p                     0.7359361      0.0081473      0.7196610      0.7515924
```

We see that the results are identical, as expected.

Handling > 1 classification variable can sometimes be a bit tricky – **but** most of the challenges are 'book-keeping'. Just remember that the dummy variable coding in the design matrix needs to be consistent with both the linear model you're trying to fit, and the structure of the `.inp` file.

6.11. Time and Group – building additive models

Most newcomers to **MARK** find building design matrices the most daunting part of the 'learning curve'. However, if you've understood everything we've covered up to now, you should find building design

matrices for additive models very straightforward. What do we mean by an 'additive model'? As you may remember from earlier chapters, an additive model is one where we express variation in survival or recapture as a function of the additive contributions of 2 or more factors.

Recall our comparison of the 'good' and 'poor' swift colonies (Chapter 4). Our linear model was represented by

$$\text{logit}(\varphi) = \beta_1 + \beta_2(\texttt{COLONY}) + \beta_3(\texttt{TIME}) + \beta_4(\texttt{COLONY.TIME})$$

In this model, we have two 'factors' – `COLONY` ('good' and 'poor') and `TIME` ($\varphi_1, \varphi_2 \ldots \varphi_7$). Each 'time interval' is considered as a different level of the `TIME` factor. In this case, we are treating time as a 'categorical' variable, as opposed to a quantitative covariate as we did in the 'trend' example presented earlier.

You may have noted that this is, in fact, the default **MARK** CJS model with 2 groups – as long as you tell **MARK** to use the same parameter structure between groups, but let the parameter values differ (i.e., same qualitative structure between the PIMs, but different indexing), then **MARK** uses the 'full model', with both factors (`COLONY` and `TIME`), and the interaction term (`COLONY.TIME`). When we run **MARK**, but set the PIMs for the 2 groups to be the same (same structure, same index values), we are testing model

$$\text{logit}(\varphi) = \beta_1 + \beta_2(\texttt{TIME})$$

The difference between these two models is, in fact, the effect of `COLONY + COLONY.TIME`, not just `COLONY` alone. As noted in Chapter 4, we are, in fact, leaving out the 'intermediate model':

$$\text{logit}(\varphi) = \beta_1 + \beta_2\texttt{COLONY} + \beta_3\texttt{TIME}$$

In this model, we are considering the additive effects of the 2 factors – hence, we refer to it as an 'additive' model. To fit this model we simply need to take our design matrix for the fully time-dependent model

$$\text{logit}(\varphi) = \beta_1 + \beta_2(\texttt{COLONY}) + \beta_3(\texttt{TIME}) + \beta_4(\texttt{COLONY.TIME})$$

and delete the interaction columns! It's that easy! In fact, we already saw this earlier when we analyzed the Dipper data with the flood model. Go back to the swift analysis, pull up the design matrix for the full model $\{\varphi_{g*t}p_{g*t}\}$, and simply delete the columns corresponding to the interaction of group and time (for survival). The design matrix (the upper-left quadrant for survival) should look like:

B1 Phi Int	B2 Phi g1	B3 Phi t1	B4 Phi t2	B5 Phi t3	B6 Phi t4	B7 Phi t5	B8 Phi t6	Parm	
1	1	1	0	0	0	0	0	1:Phi	0
1	1	0	1	0	0	0	0	2:Phi	0
1	1	0	0	1	0	0	0	3:Phi	0
1	1	0	0	0	1	0	0	4:Phi	0
1	1	0	0	0	0	1	0	5:Phi	0
1	1	0	0	0	0	0	1	6:Phi	0
1	1	0	0	0	0	0	0	7:Phi	0
1	0	1	0	0	0	0	0	8:Phi	0
1	0	0	1	0	0	0	0	9:Phi	0
1	0	0	0	1	0	0	0	10:Phi	0
1	0	0	0	0	1	0	0	11:Phi	0
1	0	0	0	0	0	1	0	12:Phi	0
1	0	0	0	0	0	0	1	13:Phi	0
1	0	0	0	0	0	0	0	14:Phi	0
0	0	0	0	0	0	0	0	15:p	1
0	0	0	0	0	0	0	0	16:p	1

What **MARK** does with this design matrix is to estimate a coefficient ('slope') for each dummy variable. This coefficient tells us how any one particular level differs from the baseline level (i.e., the last time interval). We can put all the coefficients together in one regression equation (for COLONY and TIME):

$$\text{logit}(\varphi) = \beta_1 + \beta_2(\texttt{COLONY}) + \beta_3(\texttt{t}_1) + \beta_4(\texttt{t}_2) + \beta_5(\texttt{t}_3) + \beta_6(\texttt{t}_4) + \beta_7(\texttt{t}_5) + \beta_8(\texttt{t}_6)$$

In this expression, β_1 tells us how much, on average, survival in the 'good' colony differs from survival in the 'poor' colony. Note that when COLONY=0 (say, for the 'poor' colony), the β_2 term drops out of the regression equation, resulting in our estimating survival in the 'good' colony. Each of the remaining β-terms specifies how much survival in one year period (average over both colonies) differs from the baseline year. The greater the magnitude of a particular β the greater that year's survival probability differs from the baseline year, and the greater the statistical significance of a particular β, the greater the contribution to the overall significance of the TIME factor.

It should be easy to see that, for example, for the 'poor' colony (COLONY=0) in year 3, the β_2 term drops out of the equation, and we are left with

$$\text{logit}(\varphi) = \beta_1 + \beta_5$$

because $t_3 = 1$, and all other t values ($t_1, t_2, \ldots t_6$) are equal to zero. Thus, β_5 tells us how much survival in year 3 differed from the baseline year (year 7).

Similarly, the equation for year 5 in the 'good' colony (COLONY=1) would be

$$\text{logit}(\varphi) = \beta_1 + \beta_2 + \beta_7$$

One last thing to consider. Counting parameters for the additive model (model $\{\varphi_{g+t}p_{g*t}\}$) is not quite as simple as for other models. First, we'll consider how to count the number of potentially available parameters in an additive model. The easiest way to do it is to remember what we're after – we're testing a model where there is time variation in survival for both colonies, but that the difference between colonies is due to a constant, additive, component. This additive component is simply 1 more parameter (like estimating a constant). This should be surprising, especially if you think of the additive model in the ANCOVA example we dealt with earlier – in the Lebreton *et al*. (1992) monograph, they give you an explicit 'hint' when they use the word 'parallelism'. If any pair of lines in an $X - Y$ plane are parallel then for any value X, the two lines differ in Y by some constant amount. It is the constant 'difference' between the lines which constitutes one of the estimable parameters. So, for our present example, simply count the number of parameters you would expect for 1 colony alone, for the underlying model (CJS – 13 parameters for either colony alone). Then, add 1 for the additivity (i.e., the 'constant difference') in survival (you would add 2 if you had additivity in survival and recapture simultaneously).

Now, the tricky part (potentially) – if survival in one colony is simply survival in the other colony plus a constant, then the survival probability is identifiable (by linear interpolation) for all intervals in the second colony, and thus all of the recapture probabilities are estimable (no confounding of terminal p and φ parameters). So, since there are 7 recaptures, we add 7, bringing our total to $(13 + 1 + 7) = 21$ total parameters.

Still don't get it? Here's another way to think of it. First, in this example, we are constraining the CJS model by colony. What was estimable in this starting model will remain estimable in the same model with a constraint – in this case, the additive model. Thus, if some parameters are not identifiable in the additive model, they must be those from the last time interval: $\varphi_{7,g}$ and $\varphi_{7,p}$ and $p_{8,g}$ and $p_{8,p}$ (where 'g' = good, and 'p' = poor). Let's focus our attention on them. In the unconstrained CJS model, we could

identify the products $\beta_{9,g} = \left(\varphi_{7,g} p_{8,g}\right)$ and $\beta_{9,p} = \left(\varphi_{7,p} p_{8,p}\right)$. In essence, we had 2 equations and 4 unknowns. If we pick some value for (say) $\varphi_{7,g}$, then we can solve for $p_{8,g}$. Similarly, we could pick an arbitrary value for $\varphi_{7,p}$ and solve for $p_{8,p}$. Thus, we have 2 arbitrary parameters to discount from the initial total of 28 parameters in the model – in other words, $(28 - 2) = 26$ identifiable parameters. Of course, we knew this already – we simply want to confirm that this approach yields the same results.

Now, let's apply this same line of reasoning to the additive model case. We still have the same 2 equations as before, plus 1 new one; $\varphi_{7,g} = \varphi_{7,p} + c$ (from the constraint). 'c' is a known constant, because 'c' is common to all intervals. Therefore, if we pick a value for $\varphi_{7,g}$ then $\varphi_{7,p}$ is known, and thus also both $p_{8,g}$ and $p_{8,p}$. Thus, we have just one arbitrary parameter to discount from the total number of parameters originally included in the additive model; for survival, 7 slopes + 1 intercept = 8, and for recapture, 14. Thus, $(8 + 14) = 22 - 1$ (the arbitrary parameter) = 21 identifiable parameters. However, if we run the additive model, we find that **MARK** has estimated only 19 parameters. It has 1 intercept, and 7 slopes, but of the recapture parameters, 2 are not identifiable – so $(1 + 7 + 12) = (20 - 1)$ (the arbitrary parameter) = 19.

In fact, Lebreton *et al*. (1992) analyzed a large number of different models for this data set (see Table 14). They found that the most parsimonious model has constant survival within colony, but different colony values, and additivity ('parallelism') in recapture probabilities ($\varphi_c p_{c+t}$).

Before we leave this chapter, it is worth noting that the 'parallelism' we have been discussing refers only to the logit scale – i.e., it is not *linear* parallelism, but *logit* parallelism. Thus, if you plot the reconstituted values from an additive model, they may not 'look' to be parallel, but on the logit scale, they indeed are. Suppose we had found additivity of survival between colonies for the swift data. What would this mean? It would mean that whatever factor made the survival differ overall between the colonies had a constant additive effect over time. This would imply that there is some 'real difference', possibly genetic or age, between the two colonies such that, although both of them are subject to fluctuations over time, one colony always does relatively better (or worse) than the other.

6.12. Linear models and 'effect size': a test of your understanding. . .

Recall that in chapter 4, we introduced the question of 'significance' and 'effect size'. We considered the 'swift analysis', and asked the question: is there a difference in survival between the colonies (good colony versus poor colony), and is it 'significant'? In addressing these questions, we considered the matter of 'effect size'. In the swift analysis, we concluded that there was good support for the contention that there is a difference in survival between the two colonies, based on relative AIC model weights. The remaining questions were – how big is this difference, and is the difference 'biologically meaningful'? As we note in chapter 4, the first question relates to 'effect size' – we consider colony as an 'effect', strictly analogous to an 'effect' in ANOVA. The 'effect size' is the estimate of the magnitude of the difference in survival between the two colonies. Further, since the effect size is 'estimated', it will have an associated uncertainty which we can specify in terms of a confidence interval (CI).

The key question then becomes

> *'what are the plausible bounds on the true effect size, and are biologically important effects contained within these bounds?'*

In chapter 4, we concentrated on simple interpretation of effect size. Here, we explore this a little more deeply, introducing some of the considerations involved in estimating effect size in the first place. This exercise will also serve as a test of your basic understanding of linear models (the subject of this chapter).

We'll consider a situation similar to the 'swift analysis' we introduced earlier. We'll imagine there are 2 colonies (`good` and `poor`), and that survival over a given interval in the `poor` colony is 10% lower than survival in the `good` colony over that same interval (warning: keep awake for scaling issues here!). We simulated some data, 8 occasions, 150 newly marked birds released in each colony on each occasion. We assumed a constant survival probability over time for each colony: 0.80 for the `poor` colony, and (0.80 + 10%) = 0.88 for the good colony. We also assumed a constant recapture probability of 0.75 for both colonies. The simulated data are contained in `effect_size.inp` (where the first frequency column corresponds to the `poor` colony, and the second frequency column corresponds to the `good` colony). Again, 2 groups (`poor` and `good`, 8 occasions).

While you could fit this model by (i) building the fully time-dependent model $\{\varphi_{g*t}p_{g*t}\}$ using PIMs, (ii) constructing the corresponding design matrix, and then (iii) reducing the design matrix by eliminating the columns involving `TIME` for φ, and the columns involving both `TIME` and `COLONY` for p, for this demonstration it's easier to simply start with a PIM structure that corresponds to our underlying model, $\{\varphi_g p_.\}$.

Here are the PIMs for survival

1	1	1	1	1	1	1
	1	1	1	1	1	1
		1	1	1	1	1
			1	1	1	1
				1	1	1
					1	1
						1

poor

2	2	2	2	2	2	2
	2	2	2	2	2	2
		2	2	2	2	2
			2	2	2	2
				2	2	2
					2	2
						2

good

and recapture

3	3	3	3	3	3	3
	3	3	3	3	3	3
		3	3	3	3	3
			3	3	3	3
				3	3	3
					3	3
						3

poor

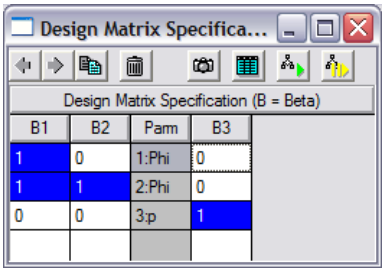
3	3	3	3	3	3	3
	3	3	3	3	3	3
		3	3	3	3	3
			3	3	3	3
				3	3	3
					3	3
						3

good

For this analysis, we're primarily interested in estimating the 'effect size' – effect of colony on survival. How do we do that? The answer in this case is fairly easy if you consider the linear model corresponding to this particular analysis. We have 2 groups (`COLONY`), and 8 occasions (corresponding to 7 `TIME` intervals). Thus, our linear model for survival would be:

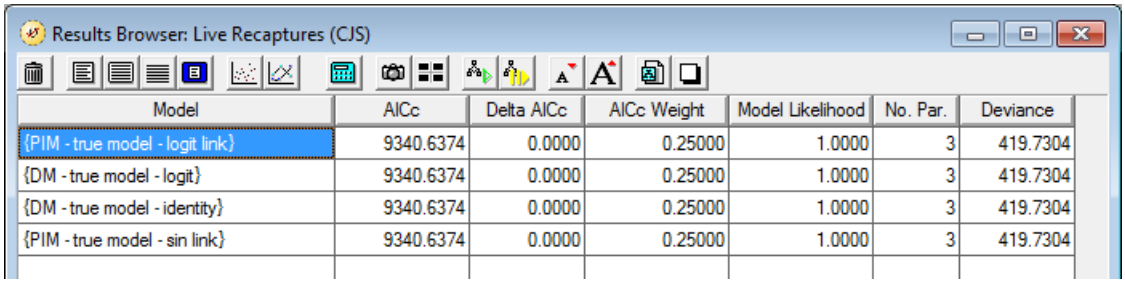
$$\text{logit}(\varphi) = \beta_1 + \beta_2(\texttt{COLONY})$$

If you've read this far, and understood the theory behind linear models, you'll recall that β_1 and β_2 together code for the colony effect. How this coding works depends on the design matrix used. If you use the design matrix coding shown at the top of the next page



then how do we interpret the intercept (β_1) and the first slope (β_2)? If the `poor` colony is coded in the 2nd column (`B2` column) of the design matrix as '`0`', and we use '`1`' for the `good` colony, then if the colony is `poor`, the intercept gives the survival for the `poor` colony (since the β_1 term drops out of the equation). So, if the intercept is the estimate for the `poor` colony, then when the dummy variable is 1 (specifying a `good` colony), then $\beta_1 + \beta_2$ = (`poor`) + (`good-poor`) = `good`. Clearly – the β_2 value is the *effect* of colony – it is the degree to which the `poor` colony differs from the `good` colony. So, the estimate for β_2 is the estimate of the effect size.

How do we actually get an estimate of the effect size on the familiar probability scale (i.e., in the range [0, 1])? In fact, we can do this in a couple of ways. Let's start with the direct way, using the identity link function. Recall that in many (perhaps most) cases, we fit models using either the logit or sin link functions. For now, though, let's re-run our model, using the identity link, which you select during the numerical estimation part of the run. Our general starting model will be $\{\varphi_{colony} p_{.}\}$. Go ahead and run it using either the default sin link, or the logit link. Then, run the model a second time using the identity link.



Note for these data that the model fit is identical regardless of whether you use the logit, sin or identity link (although by now you probably appreciate that this is not always the case).

OK, on to the next step. We want the estimate for β_2 – the slope for the colony term in the design matrix, which we understand to be the effect size – in this case, the difference between the `good` and `poor` colonies. All you need to do is look at the 'beta estimates' for the model with the identity link. The estimated value for $\hat{\beta}_2 = 0.0845106$, with a 95% CI of [0.0635335, 0.10548780]. For completeness, the estimate of the intercept is $\hat{\beta}_1 = 0.7897399$. So, the linear model is

$$\begin{aligned} \text{'survival'} = \text{logit}(\hat{\varphi}) &= \hat{\beta}_1 + \hat{\beta}_2(\text{'colony'}) \\ &= 0.7897399 + 0.0845106(\text{'colony'}) \end{aligned}$$

Now, what do these numbers tell us? Well, recall that the estimate for $\hat{\beta}_1$ is the estimate of survival for the `poor` colony. Look back at the parameter values used in the simulation for the `poor` colony: the

'true' value for the survival probability for the `good` colony is 0.8 – our estimated value for the intercept, $\hat{\beta}_1 = 0.79$, is very close. What about effect size? Well a 10% difference in survival corresponds to an absolute difference of 0.08 (since 0.8 is 10% smaller than 0.88, the true survival probability of the good colony). Since β_1 corresponds to the `poor` colony, then β_2 is the deviation ('effect') of the `good` colony on survival – in this case, $\hat{\beta}_2 = 0.0845106$ (the positive sign indicating an increase in survival for the `good` colony, relative to the `poor` colony). The estimate of $\hat{\beta}_2 = 0.0845106$ is quite close to the true difference (effect size) of 0.08 (the true difference clearly falls within the 95% CI for the estimate). Recall that these are simulated data, so we anticipate some difference between estimated and 'true' parameter values.

OK – so the estimate of the effect of colony on survival is $\hat{\beta}_2 = 0.0845106$. Is this 'significant'? The more appropriate phrasing of the question is, again:

> *'what are the plausible bounds on the true effect size, and are biologically important effects contained within these bounds?'*

The 95% CI for the estimate of the effect size ranges approximately from 0.064 to 0.105. Since this 95% CI doesn't bound 0, then we can conclude that there is a '*statistically* significant' effect of colony on survival. But, as every experienced analyst knows, if you have a big enough sample size, even minute differences can be 'statistically significant'. What about '*biologically* significant' – isn't that more relevant than 'statistical' significance? Here is where 'biological insight' comes into play. Whether or not the effect estimated in this study is 'significant or not' depends on your thoughts on what is or is not 'biologically significant'. Suppose, for example, that you decide *a priori* that a difference in survival between the colonies of $\geq$ 10% to be 'biologically significant', then in this case, our results would be considered as 'biologically inconclusive' (despite being 'statistically significant'), since the 95% CI includes values < 10%. If instead we believed that a difference in survival of 5% was 'biologically important', then we could conclude with 95% confidence that in this case there was a biologically significant result, since the 95% CI do not include this value (since the lower CI is > 5%).

Some additional comments. In the preceding, we used the identity link. We did so for convenience – the identity link gave us an estimate of the absolute value of the effect size directly, on the $[0, 1]$ probability scale we're typically interested in. But, what if the numerical estimation 'doesn't work' with the identity link (typically, because of difficulties with convergence)? Can we use the logit or sin link to get estimates of the effect size, and the standard error? The answer is 'yes', but it does require a bit more work. Let's run the analysis of the simulated data using the logit link. The estimates for $\hat{\beta}_1$ and $\hat{\beta}_2$ are 1.3233584 and 0.6157168, respectively. Remembering that the linear model we're fitting is

$$\text{logit}(\hat{\varphi}) = \hat{\beta}_1 + \hat{\beta}_2(\texttt{colony})$$

then since β_1 = (`poor`), and $\beta_1 + \beta_2$ = (`poor` + *effect* of `good`), we can write

$$\begin{aligned}\text{effect of 'good'} &= \left(\frac{e^{\hat{\beta}_1+\hat{\beta}_2(1)}}{1+e^{\hat{\beta}_1+\hat{\beta}_2(1)}}\right) - \left(\frac{e^{\hat{\beta}_1}}{1+e^{\hat{\beta}_1}}\right) \\ &= 0.8742505 - 0.789740 \\ &= 0.0845106\end{aligned}$$

which is precisely the same value as the one estimated (for β_2) using the identity link.

What about the SE of the estimated effect size? As noted earlier, the discussion of effect size is really a discussion of whether or not the confidence limits on the effect size bound a difference that we think is 'biologically significant'. From the identity link analysis, we know that the SE and confidence limits to our effect size are 0.0107 and $[0.0635, 0.1055]$, respectively.

We can derive the same values using the estimates from the logit link analysis, but it requires a few more steps. Using the logit link, we can derive an estimate of the SE as follows.

First, recall that the variance of a difference (of say two parameters θ_i and θ_j) is

$$\mathrm{var}(\theta_i - \theta_j) = \mathrm{var}(\hat{\theta}_i) + \mathrm{var}(\hat{\theta}_j) - 2\,\mathrm{cov}(\hat{\theta}_i, \hat{\theta}_j)$$

Thus, the estimated SE for the difference (i.e., effect size) between the 'good' and 'poor' colony is

$$\sqrt{\mathrm{var}(\mathtt{good}) + \mathrm{var}(\mathtt{poor}) - 2\mathrm{cov}(\mathtt{good},\mathtt{poor})}$$

where the variance and covariances of the two estimates can be output directly in **MARK**. To do this, simply select the appropriate model in the results browser (in this case, the logit link model). Then, select '**`Output | Specific Model Output | Variance-Covariance matrices | Real Estimates`**', and then output to a Dbase file (to maintain full numerical precision). The variance-covariance values for the estimates of interest (i.e., good and poor) on the real probability scale are: $\widehat{\mathrm{var}}(\mathrm{poor}) = 0.00007377$, $\widehat{\mathrm{var}}(\mathrm{good}) = 0.00004534$, and $\widehat{\mathrm{cov}}(\mathrm{poor,good}) = 0.0000022834$. Thus, our estimate of the SE for the effect size is

$$\sqrt{0.00004534 + 0.00007377 - 2(0.0000022834)} = 0.0107$$

which is the same as the estimate using the identity link (to within rounding error). The estimated 95% CI would then be (effect size±2SE), or $0.0846 \pm 2(0.0107) = [0.0632, 0.1060]$, which is virtually identical to the estimated 95% CI derived using the identity link (to within rounding error).

begin sidebar

Variance of a difference – say what??

Where does this formula for the variance of a sum, or a difference, come from? Well, if $A = X_1 + X_2$ for example, then

$$\begin{aligned}
s_A^2 &= \frac{1}{n}\sum(A-\bar{A})^2 = \frac{1}{n}\sum\left[(X_1+X_2) - \frac{1}{n}\left(X_1+X_2\right)\right]^2 \\
&= \frac{1}{n}\sum\left[(X_1+X_2) - \frac{1}{n}\sum(X_1) + \frac{1}{n}\sum(X_2)\right]^2 = \frac{1}{n}\sum\left[(X_1+X_2) - \bar{X}_1 - \bar{X}_2\right]^2 \\
&= \frac{1}{n}\sum\left[(X_1-\bar{X}_1) + (X_2-\bar{X}_2)\right]^2 = \frac{1}{n}\sum\left[x_1+x_2\right]^2 \\
&= \frac{1}{n}\sum\left[x_1^2 + x_2^2 + 2x_1x_2\right] = s_1^2 + s_2^2 + 2s_{12} \\
&= \mathrm{var}(X_1) + \mathrm{var}(X_2) + 2\mathrm{cov}(X_1, X_2)
\end{aligned}$$

Similarly, if $D = (X_1 - X_2)$ (i.e., a difference rather than a sum), then

$$s_D^2 = \mathrm{var}(X_1) + \mathrm{var}(X_2) - 2\mathrm{cov}(X_1, X_2)$$

Bet you're glad you asked!

Well, now that we've impressed ourselves, a more intuitive explanation. You may recall that the variance of a sum is equivalent to the sum of the variances, *if the elements are independent* (you should be able to prove this to yourself fairly easily).

We can write this as

$$\mathrm{var}\left(\sum \hat{x}_i\right) = \sum \mathrm{var}\left(\hat{x}_i\right)$$

But, if there is any sampling covariance (i.e., if the terms are dependent), then we write

$$\mathrm{var}\left(\sum \hat{x}_i\right) = \sum \mathrm{var}(\hat{x}_i) + \sum_i \sum_j \mathrm{cov}(\hat{x}_i, \hat{x}_j)$$

Thus, 'intuitively', given two values x_i and x_j, we write

$$\mathrm{var}(\hat{x}_i + \hat{x}_j) = \mathrm{var}(\hat{x}_i) + \mathrm{var}(\hat{x}_j) + 2\,\mathrm{cov}(\hat{x}_i, \hat{x}_j)$$

or, equivalently for a difference,

$$\mathrm{var}(\hat{x}_i - \hat{x}_j) = \mathrm{var}(\hat{x}_i) + \mathrm{var}(\hat{x}_j) - 2\,\mathrm{cov}(\hat{x}_i, \hat{x}_j)$$

We can also derive this expression using the *Delta method* (Appendix B) applied to the difference between two variables. Again, if $D = (X_1 - X_2)$, then we can show that if the variances of the X_i are small, then to first-order

$$\mathrm{var}(\mathbf{Y}) = \mathbf{D}\Sigma\mathbf{D}^{\mathbf{T}}$$

where $\mathbf{D}$ is a row-vector of the partial derivative of the function (in this case, $D = X_1 - X_2$) with respect to each variable (i.e., X_1 and X_2, respectively), $\mathbf{D}^{\mathbf{T}}$ is the column-vector transpose of $\mathbf{D}$, and Σ is the variance-covariance matrix of X_1 and X_2. The concepts underlying this expression are presented in detail in Appendix B.

Thus,

$$\begin{aligned}
\mathrm{var}(\mathbf{D}) &= \mathbf{D}\Sigma\mathbf{D}^{\mathrm{T}} \\
&= \begin{bmatrix} \dfrac{\partial D}{\partial X_1} & \dfrac{\partial D}{\partial X_2} \end{bmatrix} \begin{bmatrix} \mathrm{var}(X_1) & \mathrm{cov}(X_1, X_2) \\ \mathrm{cov}(X_2, X_1) & \mathrm{var}(X_2) \end{bmatrix} \begin{bmatrix} \dfrac{\partial D}{\partial X_1} \\ \dfrac{\partial D}{\partial X_2} \end{bmatrix} \\
&= [1 \quad -1] \begin{bmatrix} \mathrm{var}(X_1) & \mathrm{cov}(X_1, X_2) \\ \mathrm{cov}(X_2, X_1) & \mathrm{var}(X_2) \end{bmatrix} \begin{bmatrix} 1 \\ -1 \end{bmatrix} \\
&= \mathrm{var}(X_1) + \mathrm{var}(X_2) - 2\mathrm{cov}(X_1, X_2)
\end{aligned}$$

which is **identical** to the expression we introduced on the preceding page.

end sidebar

Now, let's consider a slightly more complex example – same basic scenario, but now with 3 colonies, instead of 2. Again, we simulate a data set (`effect_size3.inp`) with 3 colonies. Assuming constancy of survival over time for all 3 colonies, but let the survival differ among the colonies: for colony 1, 0.65, for colony 2, survival is 10% higher (i.e., 0.715), and in colony 3, survival is 15% higher than in colony 1 (i.e., 0.7475). Thus, colony 3 has a survival probability that is 4.55% higher than colony 2. Please note the scale – we're speaking of differences in terms of percentages – not arithmetic differences. So, for example, is a 10% difference in survival between colony 1 and colony 2 biologically meaningful? You need to think carefully about scaling, since a 10% increase in survival from a reference value of 0.5 (0.55) is different (arithmetically) than a 10% increase from a reference value of (say) 0.7 (0.77). The arithmetic difference is 0.05 in the first case, and 0.07 in the second case. However, the effect size we're

working with (as in the preceding example) is on the 'real' scale – it is not proportional (or, in terms of percent differences). So, for the present example, the effect (difference) between colony 1 and colony 2 is $(0.715 - 0.650) = 0.065$, between colony 1 and colony 3 is $(0.7475 - 0.65) = 0.0975$, and between colony 2 and colony 3 is $(0.7475 - 0.715) = 0.0325$. We set $p = 0.75$, and released 500 individuals on each occasion, for 8 occasions.

Let's see if we can derive estimates for the effect sizes. We'll save ourselves a few steps by simply going ahead and fitting the true model, which is $\{\varphi_{colony} p_{.}\}$. If we do this using the PIM chart approach (the quickest way to fit this model), we get estimates of survival of 0.6600 for colony 1, 0.7211 for colony 2, and 0.7447 for colony 3, which are all fairly close to the 'true' values specified in the simulation. So, if we wanted to quickly derive estimates of the effect size, we have to do no more than pull out our calculator, and take the pair-wise differences among these 3 values: so the effect size between colony 1 and colony 2 is $(0.7211 - 0.6600) = 0.0611$, between colony 1 and colony 3 is $(0.7447 - 0.6600) = 0.0847$, and between colony 2 and colony 3 is $(0.7447 - 0.7211) = 0.0236$. Again, these are all close to the 'true' differences expected given the values using in the simulations.

How would we get these estimates in a more 'elegant' fashion (i.e., other than by simply calculating the arithmetic difference)? Well, first, we need to specify the model using a design matrix. But, as we've seen earlier, there are a number of ways in which such a design matrix could be constructed. In this case, we're interested in looking at the magnitude of differences among levels of an effect. This is most easily done using an intercept-based model (remember the preceding example where the effect size was measured as the relative difference between the intercept term and the colony factor term in the linear model). In this example, we have 3 levels of the 'colony' effect, so we need 2 columns of dummy variables to code for this. Thus, our linear model would have an intercept term, plus 2 other terms to specify the colony. This much should be familiar (given that you've made it this far in the chapter). However, how should you code the different colonies? The following 3 design matrices (for the survival parameter; for now, we'll ignore recapture rate) are all equivalent in terms of the 'results' (i.e., the colony-specific estimates of survival), but have different interpretations:

β_1	β_2	β_3
1	0	0
1	1	0
1	0	1

β_1	β_2	β_3
1	1	0
1	0	0
1	0	1

β_1	β_2	β_3
1	1	0
1	0	1
1	0	0

The differences among these design matrices have to do with what colony is used as the 'reference' or 'control' colony. In the left-most matrix, the design matrix corresponds to a coding scheme wherein if both β_2 and β_3 are 0, then the intercept corresponds to colony 1. Why? Well, here you need to remember that in the `.INP` file, colony 1 was the first group, colony 2 was the second group, and colony 3 was the third group. In effect, each row in the design matrix corresponds to each of the different colonies. Thus, if both β_2 and β_3 are 0, then the intercept corresponds to colony 1, and as such, both colony 2 and colony 3 are then estimated 'relative' to colony 1 (remember the basic structure of the linear model: intercept + effect terms). Thus, in this case, colony 1 (given by the intercept) is the 'reference' colony – in fact, this sort of design matrix coding is often referred to as 'reference cell' coding, since whichever level of a given treatment is coded by 'all zeros' is the 'reference' (or control) level of the treatment. Got it? If so, then you should see that for the middle matrix, where the row representing colony 2 (the second row) is the reference colony. Finally, in the right-most matrix, colony 3 is the reference colony. Make sure you see this. Note that there is an interaction of the design matrices, and the order in which groups are presented in the `.INP` file.

Why do we care in this case? We care because the effect size in the linear model is calculated relative to the reference level – in this example, relative to the reference colony. Let's see how this works. We'll

start by fitting the data to our model, using a design matrix with colony 1 designated as the reference colony (i.e., making use of the structure in the left-most of the 3 example matrices noted above). Since by now you can probably do this in your sleep (hopefully a state you are not in at this stage), we'll jump right to the 'results'. Using the logit link (the default link whenever the design matrix is modified from the identity matrix), the 'beta' estimates are: $\hat{\beta}_1 = 0.6635$, $\hat{\beta}_2 = 0.2863$, and $\hat{\beta}_3 = 0.4070$. The signs of the estimates indicate that the 'effect' for colony 2 and colony 3 are both positive with respect to colony 1 (the reference colony given our design matrix). Thus, the estimate of survival for both colony 2 and colony 3 are both expected to be bigger than for colony 1 (which is exactly what we expect). What about the estimates of effect size? Well, here, we need to be somewhat careful. First, recall that if both β_2 and β_3 are 0, then the intercept refers to colony 1. Thus, the effect size between colony 2 and colony 1 is 'colony 2' - 'colony 1' = $(\beta_1 + \beta_2) - (\beta_1)$. Why does $(\beta_1 + \beta_2)$ correspond to colony 2? Note that there is no β_3 term – indicating that $\beta_3 = 0$. Thus, if $\beta_3 = 0$, but both β_1 and β_2 are not 0, then this refers to colony 2.

OK, so the estimate of the effect size for the difference between colony 2 and colony 1, back-transformed from the logit scale, is

$$\frac{e^{\hat{\beta}_1+\hat{\beta}_2}}{1+e^{\hat{\beta}_1+\hat{\beta}_2}} - \frac{e^{\hat{\beta}_1}}{1+e^{\hat{\beta}_1}} = (0.7211 - 0.6600) = 0.0610$$

which is precisely what we calculated 'by hand' from the real estimates of survival for both colonies. We could do the same thing to calculate the difference between colony 1 and colony 3 (try it as a test of your understanding – the effect size (difference) should be 0.0942). But, what about the difference between colony 2 and colony 3? Well, there are a couple of approaches. First, you could change the design matrix, setting colony 2 or colony 3 as the reference, and then using the same approach as just described, except that you have a different reference colony. But, in fact, you don't need to go to all that trouble; simply remember that colony 2 is $(\beta_1 + \beta_2)$ and that colony 3 is $(\beta_1 + \beta_3)$. Thus, the difference between colony 2 and colony 3 is

$$\frac{e^{\hat{\beta}_1+\hat{\beta}_2}}{1+e^{\hat{\beta}_1+\hat{\beta}_2}} - \frac{e^{\hat{\beta}_1+\hat{\beta}_3}}{1+e^{\hat{\beta}_1+\hat{\beta}_3}} = (0.2553 - 0.2789) = -0.0236$$

So, estimated survival for colony 2 is -0.0236 lower than estimated survival for colony 3 – exactly what we 'calculated by hand' using the real estimates of survival for colonies 2 and 3. So, obviously, you can get the estimate of 'effect size' right from the 'real estimates', without going through all the effort of getting the β estimates, and back-transforming the equation (as we have done here).

But, what about the SE for the estimates of effect size? Again, since we're dealing with the variance (or SE) of a difference (in this case, between any pair of colonies), we simply output the variance-covariance values for the real estimates. For this example, the variance for β_1 is 0.00004, for β_2 is 0.00003, and for β_3 is 0.00003. Since each of the colonies is 'independent' of each other, we don't anticipate any sampling covariance – this is what we see: Cov(colony 1, colony 2) = Cov(colony 1, colony 3) = Cov(colony 2, colony 3) = 0. Thus, given the variances, the SE for the difference between colony 1 and colony 2 (for example) is

$$\sqrt{(0.00004 + 0.00003 - 0)} = 0.0084$$

And thus the approximate 95% CI for the effect (difference) between colony 1 and colony 2 is [0.0442, 0.0778]. In fact, if you fit these data using the identity link, you'll see that this estimated SE matches that given by the identity link almost exactly (remember – while the identity link is generates

estimates of the effect size and the SE directly on the normal[0, 1] probability scale, not all data sets can be fit using the identity link – usually due to convergence issues. In such cases, a transformation – like a logit or sin transform – is required).

So, we see that we can fairly easily come up with estimates of the effect size, and the SE of these estimates. We leave it to you to tackle the more difficult (at least conceptually) question of what constitutes a 'biologically meaningful' effect size. However, that 'debate' notwithstanding, we suggest that you routinely consider effect size where possible.

6.12.1. Linear models: β estimates and odds ratios

The β terms in the linear model are interpretable as the natural log of the *odds ratios*. The odds of 'success' (e.g., survival, movement) is the ratio of the probability of 'success' (say, θ, where θ is some [0, 1] bounded parameter) to the probability of 'failure' (given by the complement, $1 - \theta$). Note we assume binary states – success or failure (live or die, move or stay...).

In other words, the 'log odds of success' is given by

$$\ln\left(\frac{\theta}{1-\theta}\right)$$

The *odds ratio* between (say) two levels of some classification (treatment) variable, would be

$$\ln\left(\frac{\theta_1/(1-\theta_1)}{\theta_2/(1-\theta_2)}\right)$$

As written, we see that the odds ratio is a way of comparing whether the probability of a certain event is the same for two groups. A log odds-ratio of 0 implies that the event is equally likely in both groups. A log odds-ratio > 0 implies that the event is more likely in the first group. Conversely, a log odds-ratio < 0 implies that the event is more likely in the second group.

Consider the following example – males and females marked with a radio-telemetry device, which allows us to know the fate of the marked individual with certainty (known-fate analysis is covered in detail in chapter 16). Suppose we mark 60 males, and 45 females. Over the sampling interval, 9 males and 11 females died. Thus, the estimated survival for males is $\hat{S}_m = (51/60) = 0.85$, and for females is $\hat{S}_f = (34/45) = 0.756$. Thus, the log odds of a male surviving the interval is $\ln(0.85/0.15) = 1.735$, and the log odds of a female surviving the interval is $\ln(0.756/0.244) = 1.131$. The change in the odds ratio of survival for the two sexes is

$$\ln\left(\frac{\hat{S}_m/(1-\hat{S}_m)}{\hat{S}_f/(1-\hat{S}_f)}\right) = \ln\left(\frac{0.85/0.15}{0.756/0.244}\right) = \ln(1.829) = 0.604$$

OK, fine, but how do log odds-ratios connect to the β estimates in a linear model? Let's revisit the 'good colony' versus 'poor colony' analysis we introduced earlier in this section. In that example, we simulated data where survival in the 'good' colony was 0.88, and 0.8 in the 'poor' colony (a 10% difference). Assuming a time-invariant model (which in fact was the true model under which the data were simulated), this corresponds to an *expected* log odds of survival in the 'good' colony of $\ln(0.88/0.12) = 1.992$, and an *expected* log odds of survival in the 'poor' colony of $\ln(0.80/0.20) = 1.386$.

Recall that we fit the following linear model to those simulated data:

$$\text{logit}(\varphi) = \beta_1 + \beta_2(\texttt{COLONY})$$

The actual *estimates* for $\hat{\beta}_1$ and $\hat{\beta}_2$ were 1.3234 and 0.6157, respectively. Since β_1 = (poor), then the *estimated* log odds for survival in the 'poor' colony is simply 1.3234 (remember, this parameter is estimated on the logit scale), which is fairly close to the expected value of 1.386. For the 'good' colony, the *estimated* log-odds ratio for survival is $(\hat{\beta}_1 + \hat{\beta}_2) = (1.3234 + 0.6157) = 1.9391$, which again is quite close to the *expected* value of 1.992.

Final step – the *expected* log odds difference between the two colonies is ln[(0.88/0.12)/(0.80/0.20)] = ln(1.833) = 0.606. Since β_1 = (poor), and $\beta_1 + \beta_2$ = (poor + *effect* of good), then β_2 is the effect of the 'good' colony. Notice that $\hat{\beta}_2 = 0.6157$, which is close to the expected log-odds ratio of 0.606. Coincidence? No! The natural log of the odds ratio between the 'good' and 'poor' colonies is a measure of the difference between the two colonies – i.e., the effect size. Since the log odds ratio in this example is >0, we conclude that the event (survival) is more likely in the 'good' colony (numerator) than in the 'poor' colony (denominator).

begin sidebar

AIC, P-Values and effect size – a tautology?

Hmmm...you might suspect that you smell a tautology here (it's either that, or the aroma of your frying brain cells). Up until now, we've considered the use of AIC as a robust means of model selection – part of the motivation being to avoid the use (and abuse) of classical '*P*-value' approaches to doing science. While there are very good motivations for doing this (see relevant sections of the Burnham & Anderson text), one of the motivations was to force us (biologists, analysts) to focus our attention more on the 'biological significance' of the effect size, rather than on some nominal '*P*-value'. We all know that it is not hard to generate a situation where we can show 'statistical significance' that has no biological meaning (this commonly occurs with very large data sets). So, we consider the effect size. Recall that our purpose is

'what are the plausible bounds on the true effect size, and are biologically important effects contained within these bounds?'

The potential problem is with the word 'plausible'. In theory, we are supposed to decide, *a priori*, on what the biologically plausible bounds are. And yet, in practice, to determine whether or not a calculated effect size falls within those bounds is based on assessing the confidence bounds estimated for the effect size, relative to the plausibility bounds designated by the scientist. Herein is the problem – we use 'biological insight' to determine what we think a plausible (or biologically meaningful) effect should be, and yet we end up relying on use of 95% CI to test whether or not the effect size is plausible. And what do we base the 95% CI on? You got it – a nominal $\alpha = 0.05$ value. Remember, that it is in part the arbitrariness of selecting the appropriate α level which underlies one of the several criticisms of the '*P*-value' approach. And yet, we potentially use the same underlying logic to derive the 95% CI for the effect size. There is good reason to wonder if we're not engaged in some sort of circular thinking here...stay tuned.

end sidebar

6.12.2. $\hat{c}$ and effect size: a cautionary note

In the preceding, we illustrate the basic idea (and some of the mechanics) for estimating effect size. As we noted in Chapter 4, model selection (whether you use an information theoretic approach based on AIC, or the more traditional approach based on LRT) is conditional on adequate fit of the model to the data. Some degree of lack of fit can be accommodated using a quasi-likelihood approach. This involved estimation of c, perhaps by using a bootstrap, for example. However, one thing which may not be immediately obvious is the effect that using a quasi-likelihood adjustment has on model selection.

Consider, for example, what adjusting model fit using a $\hat{c} > 1$. One of the things you'll quickly notice if you progressively experiment by increasing $\hat{c}$ from 1, 1.25, 1.5, and so on, is that as you do so, the rankings of the models changes. Invariably, as $\hat{c}$ increases, models with fewer parameters take on progressively more support in the data, as indicated by the normalized AIC weights. This should make some intuitive sense: a large value of $\hat{c}$ indicates significant lack of fit of the data to the general model. As such, increasing $\hat{c}$ is analogous to 'taking a more conservative' view of the data. The general model doesn't fit, so you tend to favor the more parsimonious model. Try it – pick a model set, and slowly, progressively, try increasing $\hat{c}$ from the default of 1. Watch closely to see how the model weights change, such that (typically) reduced parameter models get increasing amounts of weight.

However, what is not so intuitive is that changing $\hat{c}$ also changes the relative scaling of differences in model support. In short form, if 2 models differ in normalized AIC weights by some factor x for $\hat{c} = 1$, then this same difference x is not the same difference if $\hat{c} > 1$; it is less. For example, suppose you have 2 models, with $\hat{c} = 1.0$,where the difference in relative weight is (say) 2.5 times (in other words, one model has 2.5 times more weight than the other model). At this point, you look at effect size, and interpret the biological plausibility of this difference, given that one model has 2.5 more support in the data. However, suppose instead that $\hat{c} = 1.5$, instead of 1. With increasing $\hat{c}$, a difference of 2.5 times support in the data is scaled differently, and must be interpreted differently, than it would be if $\hat{c} = 1$. Since increasing $\hat{c}$ increases the degree of 'conservatism' in the analysis, a difference of 2.5 times model support with $\hat{c} = 1.5$ is 'less of a difference' than the same 2.5 times difference would be with $\hat{c} = 1.0$. Confused? Fair enough – it is rather non-intuitive, at first. Give it some thought. For those who want all the details, we suggest you have a look at Richard Royall's 1997 text on '*Statistical Evidence: A Likelihood Paradigm*' (a Chapman & Hall publication – part of the Monographs series on Statistics and Applied Probability). In short – be a little cautious in how you interpret relative differences in normalized AIC weights if $\hat{c} > 1$.

6.13. Pulling all the steps together: a sequential approach

Hopefully, everything we've covered up until now makes reasonable sense. Here, we summarize the basic sequence of steps of building design matrices, and interpreting the values of the estimate β terms (the 'slopes' of the linear model).

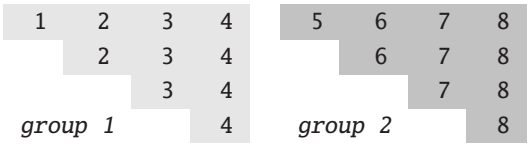
We'll introduce the approach using a very simple model: assume we have collected live mark-encounter data from 2 groups, over 4 time periods (i.e., 5 sampling occasions). To simplify the presentation somewhat, we'll focus only on the survival parameter φ (but clearly, the same basic idea holds for the recapture parameter p as well).

Step 1: decide which model you want to fit.

For now, let's assume the model we're interested in is model $\{\varphi_g\}$ – in other words, a model where survival varies only as a function of the `GROUP`, but not `TIME`.

Step 2: set up PIMs for general model

For a starting model $\{\varphi_{g*t}\}$ (i.e., a model with a full interaction of `GROUP` and `TIME`) our PIMs would look like



Step 3: set up the linear model equation for model you want to fit

We want to fit model $\{\varphi_g\}$, which we write in our symbolic linear model notation as

$$\text{logit}(\varphi) = \beta_1 + \beta_2(\texttt{GROUP})$$

Step 4: set up the design matrix

In the following, the rows of the table correspond to the PIM index values for each time interval. The design matrix corresponding to $\text{logit}(\varphi) = \beta_1 + \beta_2(\texttt{GROUP})$ is given by the columns labeled β_1 and β_2. Note that as written, we have specified GROUP 2 to be the reference group (i.e., if $\beta_2 = 0$, then the intercept – β_1 – corresponds to GROUP 2).

			INTCP	GROUP
	PIM	Param	β_1	β_2
group 1	1	$\varphi_{g1,1}$	1	1
	2	$\varphi_{g1,2}$	1	1
	3	$\varphi_{g1,3}$	1	1
	4	$\varphi_{g1,4}$	1	1
group 2	5	$\varphi_{g2,1}$	1	0
	6	$\varphi_{g2,2}$	1	0
	7	$\varphi_{g2,3}$	1	0
	8	$\varphi_{g2,4}$	1	0

Step 5: confirm the design matrix with the equations

Here, we simply take the dummy coding for each GROUP and TIME combination, and substitute into the linear model equation $\text{logit}(\varphi) = \beta_1 + \beta_2(\texttt{GROUP})$.

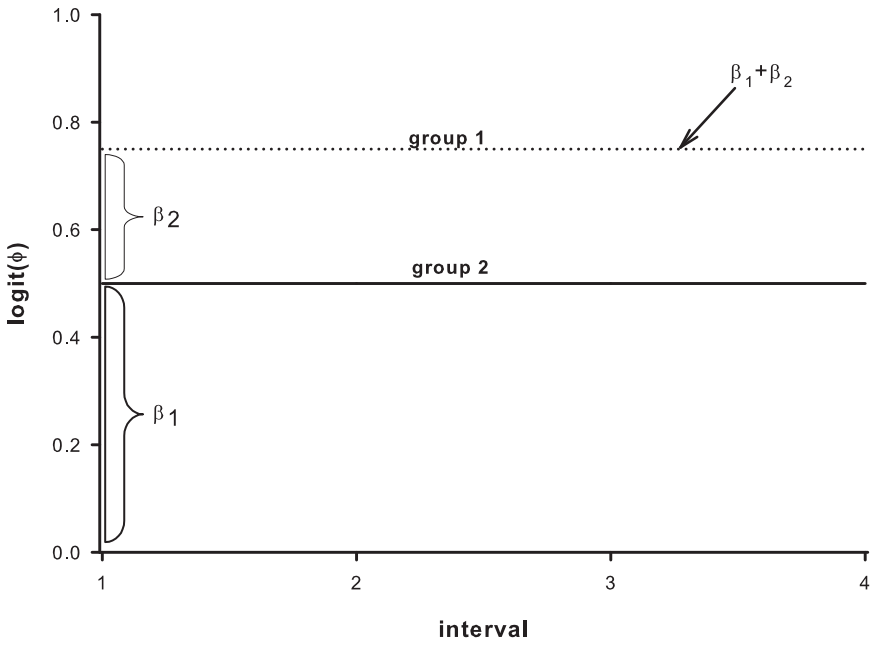
group 1	$\text{logit}(\varphi_{g1,1}) = \beta_1(1) + \beta_2(1)$	$\text{logit}(\varphi_{g1,1}) = \beta_1 + \beta_2$
	$\text{logit}(\varphi_{g1,2}) = \beta_1(1) + \beta_2(1)$	$\text{logit}(\varphi_{g1,2}) = \beta_1 + \beta_2$
	$\text{logit}(\varphi_{g1,3}) = \beta_1(1) + \beta_2(1)$	$\text{logit}(\varphi_{g1,3}) = \beta_1 + \beta_2$
	$\text{logit}(\varphi_{g1,4}) = \beta_1(1) + \beta_2(1)$	$\text{logit}(\varphi_{g1,4}) = \beta_1 + \beta_2$
group 2	$\text{logit}(\varphi_{g2,1}) = \beta_1(1) + \beta_2(0)$	$\text{logit}(\varphi_{g2,1}) = \beta_1$
	$\text{logit}(\varphi_{g2,2}) = \beta_1(1) + \beta_2(0)$	$\text{logit}(\varphi_{g2,2}) = \beta_1$
	$\text{logit}(\varphi_{g2,3}) = \beta_1(1) + \beta_2(0)$	$\text{logit}(\varphi_{g2,3}) = \beta_1$
	$\text{logit}(\varphi_{g2,4}) = \beta_1(1) + \beta_2(0)$	$\text{logit}(\varphi_{g2,4}) = \beta_1$

On the right hand side we see the result of substituting in the dummy variables. We see clearly that GROUP 2 is coded by the intercept: thus, β_2 represents the difference ('effect size') between GROUP 2 and GROUP 1.

Step 6: plot a graph to illustrate the model

Plotting a graph of the model is a very useful way to visualize (and thus, truly understand) the structure of the model, and what the individual β_i terms represent.

For this model, the graph of the β terms is shown at the top of the next page. Note that the meaning of the β_i terms is explicitly represented, as is the effect size (which, in this case, is the value of β_2, which is the difference between the two horizontal lines on the logit scale).



Got it? Well, let's test our understanding by trying several more scenarios.

Let's take the same basic data model (2 groups, 5 occasions), and now consider fitting model φ_t – `TIME` variation in φ, but no `GROUP` effect (i.e., essentially the opposite of the model we just considered).

Step 1: decide which model you want to fit.

As noted, let's consider model $\{\varphi_t\}$ – in other words, a model where survival varies only as a function of the `TIME`, but not `GROUP`.

Step 2: set up PIMs for general model

As above, our general model would still be $\{\varphi_{g*t}\}$ (i.e., a model with a full interaction of `GROUP` and `TIME`). For this model, our PIMs would look like the following

1	2	3	4		5	6	7	8
	2	3	4			6	7	8
		3	4				7	8
group 1			4		*group 2*			8

Step 3: set up the linear model equation for model you want to fit

We want to fit model $\{\varphi_t\}$, which we write in our symbolic linear model notation as

$$\text{logit}(\varphi) = \beta_1 + \beta_2(t_1) + \beta_3(t_2) + \beta_4(t_3)$$

Now, make sure you understand why this is the appropriate linear model. We have 5 sampling occasions, which means 4 intervals. To uniquely code the intervals, we need $(n - 1) = (4 - 1) = 3$ columns of dummy variables, or (more specifically) 3 $\beta_{i,i\neq1}$ terms in addition to the intercept term β_1.

Step 4: set up the design matrix

In the following table the rows correspond to the PIM index values for each time interval. The design matrix corresponding to logit(φ) = $\beta_1 + \beta_2(t_1) + \beta_3(t_2) + \beta_4(t_3)$ is given by the columns labeled $\beta_1 \rightarrow \beta_4$. Note that as written, we have specified TIME 4 (i.e., the intervals between sampling occasion 4 and 5) to be the reference group (i.e., if $\beta_2 = \beta_3 = \beta_4 = 0$, then the intercept – β_1 – corresponds to TIME 4).

			INTCPT	TIME1	TIME2	TIME3
	PIM	Param	β_1	β_2	β_3	β_4
group 1	1	$\varphi_{g1,1}$	1	1	0	0
	2	$\varphi_{g1,2}$	1	0	1	0
	3	$\varphi_{g1,3}$	1	0	0	1
	4	$\varphi_{g1,4}$	1	0	0	0
group 2	5	$\varphi_{g2,1}$	1	1	0	0
	6	$\varphi_{g2,2}$	1	0	1	0
	7	$\varphi_{g2,3}$	1	0	0	1
	8	$\varphi_{g2,4}$	1	0	0	0

Step 5: confirm the design matrix with the equations

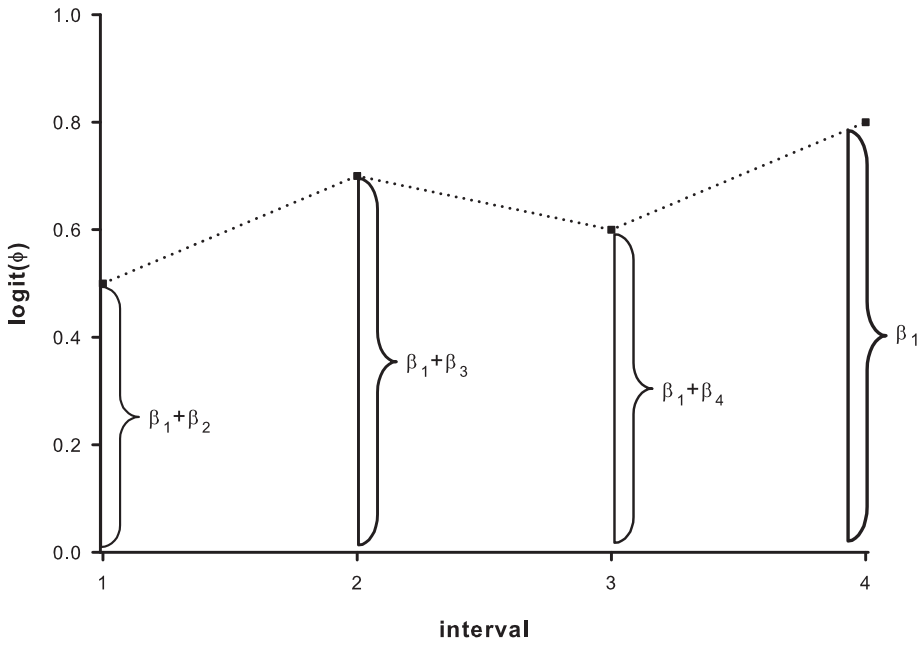
Here, we simply take the dummy coding for each GROUP and TIME combination, and substitute into the linear model equation logit(φ) = $\beta_1 + \beta_2(t_1) + \beta_3(t_2) + \beta_4(t_3)$.

group 1	$\text{logit}(\varphi_{g1,1}) = \beta_1(1) + \beta_2(1) + \beta_3(0) + \beta_4(0)$	$\text{logit}(\varphi_{g1,1}) = \beta_1 + \beta_2$
	$\text{logit}(\varphi_{g1,2}) = \beta_1(1) + \beta_2(0) + \beta_3(1) + \beta_4(0)$	$\text{logit}(\varphi_{g1,2}) = \beta_1 + \beta_3$
	$\text{logit}(\varphi_{g1,3}) = \beta_1(1) + \beta_2(0) + \beta_3(0) + \beta_4(1)$	$\text{logit}(\varphi_{g1,3}) = \beta_1 + \beta_4$
	$\text{logit}(\varphi_{g1,4}) = \beta_1(1) + \beta_2(0) + \beta_3(0) + \beta_4(0)$	$\text{logit}(\varphi_{g1,4}) = \beta_1$
group 2	$\text{logit}(\varphi_{g2,1}) = \beta_1(1) + \beta_2(1) + \beta_3(0) + \beta_4(0)$	$\text{logit}(\varphi_{g2,1}) = \beta_1 + \beta_2$
	$\text{logit}(\varphi_{g2,2}) = \beta_1(1) + \beta_2(0) + \beta_3(1) + \beta_4(0)$	$\text{logit}(\varphi_{g2,2}) = \beta_1 + \beta_3$
	$\text{logit}(\varphi_{g2,3}) = \beta_1(1) + \beta_2(0) + \beta_3(0) + \beta_4(1)$	$\text{logit}(\varphi_{g2,3}) = \beta_1 + \beta_4$
	$\text{logit}(\varphi_{g2,4}) = \beta_1(1) + \beta_2(0) + \beta_3(0) + \beta_4(0)$	$\text{logit}(\varphi_{g2,4}) = \beta_1$

On the right hand side we see the result of substituting in the dummy variables. We see clearly that TIME 4 is coded by the intercept: thus, $\beta_2 \rightarrow \beta_4$ represents the various differences ('effect sizes') between the different TIME intervals, and the final TIME interval (TIME 4).

Step 6: plot a graph to illustrate the model

Note (in the figure on the top of the next page) that the meaning of the β_i terms is explicitly represented, as are the effect sizes (which, in this case, is the value of the difference $\left(\beta_1 + \beta_{i\neq 1}\right)$ and β_1.



Now, for a final test, to really make sure you've got it. You've probably anticipated model $\{\varphi_{g*t}\}$. Here it is!

Step 1: decide which model you want to fit.

For our final example, let's consider model $\{\varphi_{g*t}\}$ – in other words, a model where survival varies as a function of the both GROUP and TIME, with full interaction between the two.

Step 2: set up PIMs for general model

Clearly, our general model must be $\{\varphi_{g*t}\}$ (i.e., a model with a full interaction of GROUP and TIME), since this is in fact the model we're trying to fit! For this model, our PIMs are, again

1	2	3	4		5	6	7	8
	2	3	4			6	7	8
		3	4				7	8
group 1			4		*group 2*			8

Step 3: set up the linear model equation for model you want to fit

We want to fit model $\{\varphi_{g*t}\}$, which we write in our symbolic linear model notation as

$$\begin{aligned}\text{logit}(\varphi) = {} & \beta_1 + \beta_2(\texttt{GROUP}) + \beta_3(\texttt{t}_1) + \beta_4(\texttt{t}_2) + \beta_5(\texttt{t}_3) \\ & + \beta_6(\texttt{GROUP.t}_1) + \beta_7(\texttt{GROUP.t}_2) + \beta_8(\texttt{GROUP.t}_3)\end{aligned}$$

Now, make sure you understand why this is the appropriate linear model. We have 2 groups, and 5 sampling occasions, which means 4 intervals. To uniquely code the for intervals, we need $(n - 1) =$

$(4-1) = 3$ columns of dummy variables for TIME, $(n-1) = (2-1) = 1$ column for GROUP, and $(3 \times 1) = 3$ columns for the interaction of GROUP.TIME. So, a total of $(1+3+1+3) = 8$ columns (β_i terms).

Step 4: set up the design matrix

In the following, the rows of the table correspond to the PIM index values for each time interval. The design matrix corresponding to

$$\begin{aligned}\text{logit}(\varphi) = {} & \beta_1 + \beta_2(\texttt{GROUP}) + \beta_3(\texttt{t}_1) + \beta_4(\texttt{t}_2) + \beta_5(\texttt{t}_3) \\ & + \beta_6(\texttt{GROUP.t}_1) + \beta_7(\texttt{GROUP.t}_2) + \beta_8(\texttt{GROUP.t}_3)\end{aligned}$$

is given by the columns labeled $\beta_1 \rightarrow \beta_8$. Note that as written, we have specified GROUP 2 during TIME 4 to be the reference group and time (i.e., if $\beta_2 = \beta_3 = \ldots \beta_8 = 0$, then the intercept – β_1 - corresponds to GROUP 2 during TIME 4).

			INTCPT	GROUP	TIME1	TIME2	TIME3	G.T1	G.T2	G.T3
	PIM	Param	β_1	β_2	β_3	β_4	β_5	β_6	β_7	β_8
group 1	1	$\varphi_{g1,1}$	1	1	1	0	0	1	0	0
	2	$\varphi_{g1,2}$	1	1	0	1	0	0	1	0
	3	$\varphi_{g1,3}$	1	1	0	0	1	0	0	1
	4	$\varphi_{g1,4}$	1	1	0	0	0	0	0	0
group 2	5	$\varphi_{g2,1}$	1	0	1	0	0	0	0	0
	6	$\varphi_{g2,2}$	1	0	0	1	0	0	0	0
	7	$\varphi_{g2,3}$	1	0	0	0	1	0	0	0
	8	$\varphi_{g2,4}$	1	0	0	0	0	0	0	0

Step 5: confirm the design matrix with the equations

Here, we simply take the dummy coding for each GROUP and TIME combination, and substitute into the linear model equation, which again is

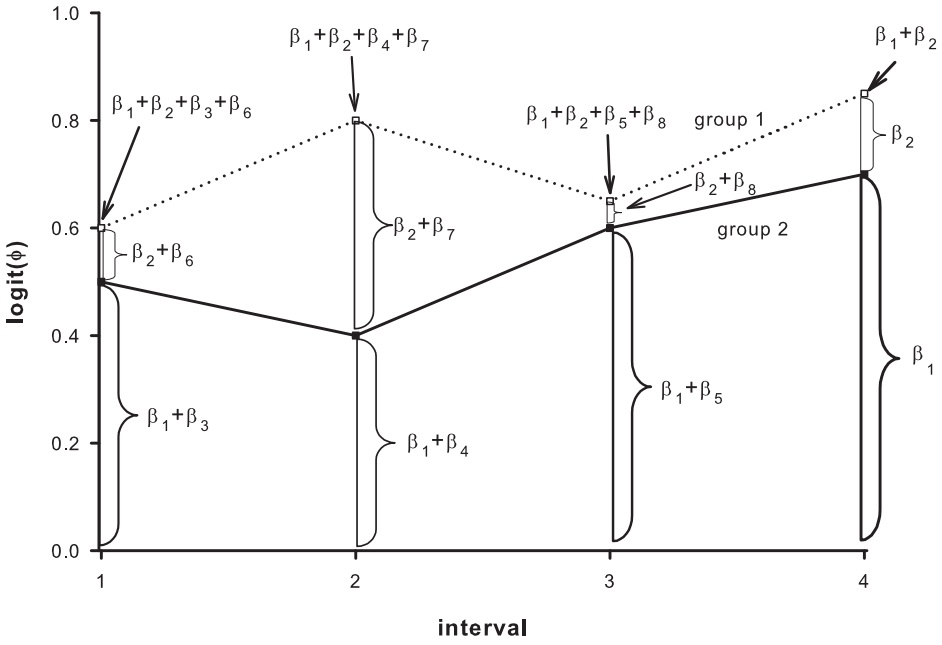
$$\begin{aligned}\text{logit}(\varphi) = {} & \beta_1 + \beta_2(\texttt{GROUP}) + \beta_3(\texttt{t}_1) + \beta_4(\texttt{t}_2) + \beta_5(\texttt{t}_3) \\ & + \beta_6(\texttt{GROUP.t}_1) + \beta_7(\texttt{GROUP.t}_2) + \beta_8(\texttt{GROUP.t}_3)\end{aligned}$$

gr 1	$\text{logit}(\varphi_{g1,1}) = \beta_1(1) + \beta_2(1) + \beta_3(1) + \beta_4(0) + \beta_5(0) + \beta_6(1) + \beta_8(0) + \beta_8(0)$	$\text{logit}(\varphi_{g1,1}) = \beta_1 + \beta_2 + \beta_3 + \beta_6$
	$\text{logit}(\varphi_{g1,2}) = \beta_1(1) + \beta_2(1) + \beta_3(0) + \beta_4(1) + \beta_5(0) + \beta_6(0) + \beta_7(1) + \beta_8(0)$	$\text{logit}(\varphi_{g1,2}) = \beta_1 + \beta_2 + \beta_4 + \beta_7$
	$\text{logit}(\varphi_{g1,3}) = \beta_1(1) + \beta_2(1) + \beta_3(0) + \beta_4(0) + \beta_5(1) + \beta_6(1) + \beta_7(0) + \beta_8(1)$	$\text{logit}(\varphi_{g1,3}) = \beta_1 + \beta_2 + \beta_5 + \beta_8$
	$\text{logit}(\varphi_{g1,4}) = \beta_1(1) + \beta_2(1) + \beta_3(0) + \beta_4(0) + \beta_5(0) + \beta_6(0) + \beta_7(0) + \beta_8(0)$	$\text{logit}(\varphi_{g1,4}) = \beta_1 + \beta_2$
gr 2	$\text{logit}(\varphi_{g2,1}) = \beta_1(1) + \beta_2(0) + \beta_3(1) + \beta_4(0) + \beta_5(0) + \beta_6(0) + \beta_7(0) + \beta_8(0)$	$\text{logit}(\varphi_{g2,1}) = \beta_1 + \beta_3$
	$\text{logit}(\varphi_{g2,2}) = \beta_1(1) + \beta_2(1) + \beta_3(0) + \beta_4(1) + \beta_5(0) + \beta_6(0) + \beta_7(0) + \beta_8(0)$	$\text{logit}(\varphi_{g2,2}) = \beta_1 + \beta_4$
	$\text{logit}(\varphi_{g2,3}) = \beta_1(1) + \beta_2(1) + \beta_3(0) + \beta_4(0) + \beta_5(1) + \beta_6(1) + \beta_7(0) + \beta_8(0)$	$\text{logit}(\varphi_{g2,3}) = \beta_1 + \beta_5$
	$\text{logit}(\varphi_{g2,4}) = \beta_1(1) + \beta_2(1) + \beta_3(0) + \beta_4(0) + \beta_5(0) + \beta_6(0) + \beta_7(0) + \beta_8(0)$	$\text{logit}(\varphi_{g2,4}) = \beta_1$

On the right hand side we see the result of substituting in the dummy variables. We see clearly that GROUP 2 at TIME 4 is coded by the intercept: thus, $\beta_2 \rightarrow \beta_8$ represents the various differences ('effect sizes') between the different TIME and GROUP combinations intervals, and GROUP 2 during TIME 4.

Step 6: plot a graph to illustrate the model

The figure for model $\{\varphi_{g*t}\}$ is shown below – pay close attention to all the interactions, and effects.



Got it? Hopefully you've now got the basic idea. Going through some version of this sequence for any model you might want to build will build understanding, and confidence.

begin sidebar

Design matrix coding and estimability

Earlier in this chapter, we noted that there are any number of ways to code the design matrix, each yielding equivalent estimates, but differing in how the individual β terms of the linear model are interpreted.

However, sometimes, there are some subtle steps to constructing a design matrix which are well worth keeping in mind. For example, consider the basic structure for the recapture part of the design matrix – up until now, we've used a reference coding scheme where we usually make the final element of the matrix the reference element.

For example, if we had the following matrix for (say) φ_t for a design with 4 time intervals

1	1	0	0
1	0	1	0
1	0	0	1
1	0	0	0

we're specifying that the last time interval is used as the reference (such that the intercept β_1 is the estimate of survival (on some scale) for this interval, and all the other β terms represent deviations from this reference.

But, suppose instead we had constructed our design matrix using

1	0	0	0
1	1	0	0
1	0	1	0
1	0	0	1

such that the estimate of survival over the first interval is now the reference value. We know from everything we've covered up until now that changing the reference coding scheme used in the design matrix will change the interpretation of the β_i terms, but does it change anything else?

The general answer is, 'no' – it shouldn't. At least, most of the time. In general, the model deviance (fit) and the reconstituted estimates should not be influenced by the choice of the coding scheme used in the design matrix. But, there are some somewhat 'pathological' cases where it might. We'll have a quick look at one example, if only to prove a point, and give you another reason to exercise your understanding of the design matrix, and how the dummy-variable coding works.

Consider the European Dipper (yes, again) – this time, using the input file containing data from both males and females (`ed.inp`). We fit model $\{\varphi_{g*t} p_{g*t}\}$ to these data, using the design matrix shown at the top of the next page (which you'll recall is the default full '`time` × `group`' design matrix in **MARK**). Look closely at the coding. We see that we've made the terminal time periods the reference cell. This is the default in **MARK** (which you can change by selecting the appropriate option under '**`File | Preferences`**').

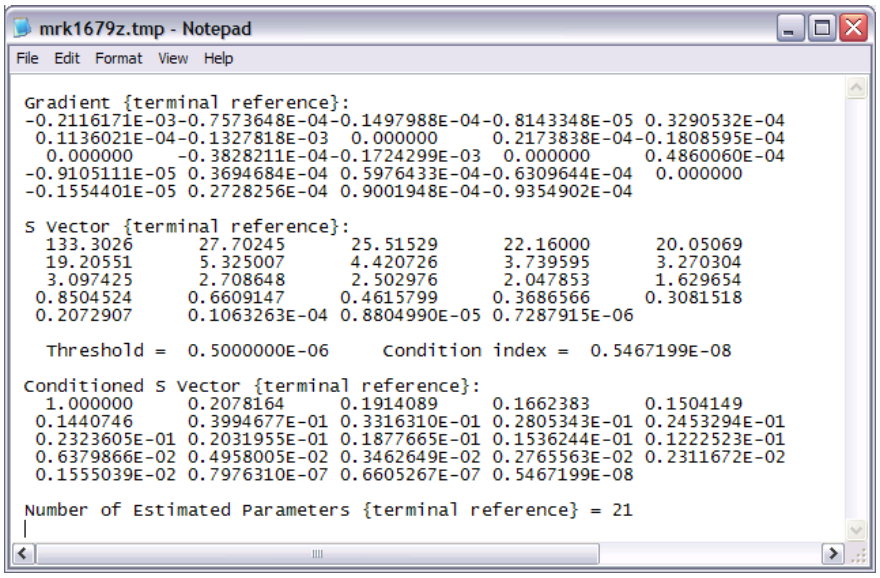
OK – so what's wrong with that? Well, perhaps nothing obvious, but what do we know about this analysis: 2 groups, 7 occasions, so 24 columns in the design matrix (as shown). But, how many parameters are estimable? Well, if you remember when we first introduced this data set, you might recall that the final $\varphi_6 p_7$ values for both sexes aren't separately identifiable (the so-called 'beta' parameters). So, $(24 - 2) = 22$ estimable parameters. But, if you run this model, using this design matrix and the logit link function, **MARK** reports only 21 parameters. Why? Because with the logit link, **MARK** was unable to estimate the second encounter probability for males ($p_{m,3}$).

Design Matrix Specification: Live Recaptures (CJS)

Design Matrix Specification (B = Beta)

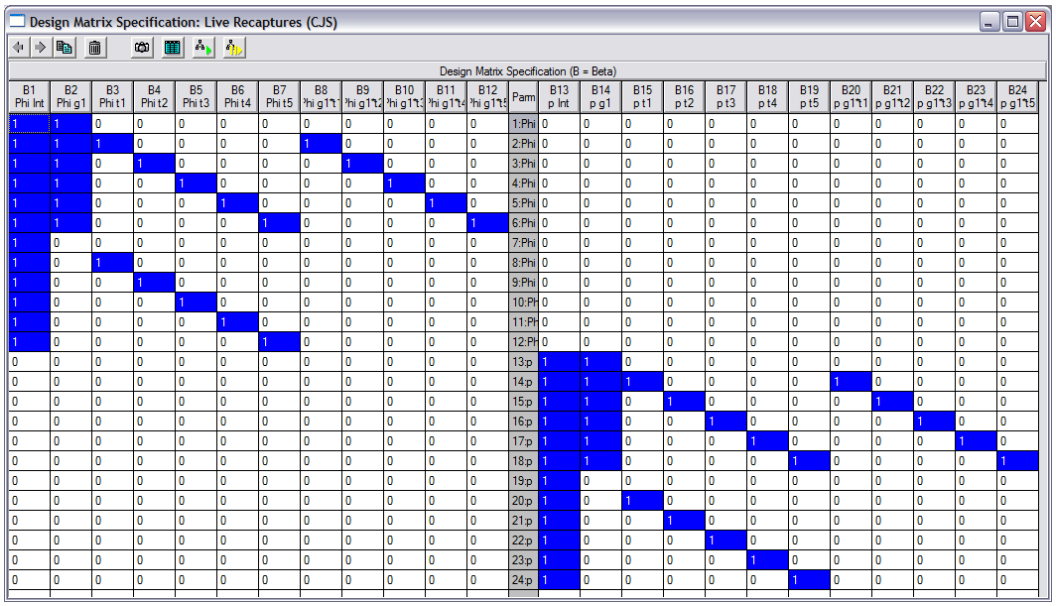
B1 Phi Int	B2 Phi g1	B3 Phi t1	B4 Phi t2	B5 Phi t3	B6 Phi t4	B7 Phi t5	B8 Phi g1t1	B9 Phi g1t2	B10 Phi g1t3	B11 Phi g1t4	B12 Phi g1t5	Parm	B13 p Int	B14 p g1	B15 p t1	B16 p t2	B17 p t3	B18 p t4	B19 p t5	B20 p g1t1	B21 p g1t2	B22 p g1t3	B23 p g1t4	B24 p g1t5
1	1	1	0	0	0	0	1	0	0	0	0	1:Phi	0	0	0	0	0	0	0	0	0	0	0	0
1	1	0	1	0	0	0	0	1	0	0	0	2:Phi	0	0	0	0	0	0	0	0	0	0	0	0
1	1	0	0	1	0	0	0	0	1	0	0	3:Phi	0	0	0	0	0	0	0	0	0	0	0	0
1	1	0	0	0	1	0	0	0	0	1	0	4:Phi	0	0	0	0	0	0	0	0	0	0	0	0
1	1	0	0	0	0	1	0	0	0	0	1	5:Phi	0	0	0	0	0	0	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0	0	0	0	6:Phi	0	0	0	0	0	0	0	0	0	0	0	0
1	0	1	0	0	0	0	0	0	0	0	0	7:Phi	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	1	0	0	0	0	0	0	0	0	8:Phi	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	0	0	0	0	9:Phi	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	1	0	0	0	0	0	0	10:Ph	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	1	0	0	0	0	0	11:Ph	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	12:Ph	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	13:p	1	1	1	0	0	0	0	1	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	14:p	1	1	0	1	0	0	0	0	1	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	15:p	1	1	0	0	1	0	0	0	0	1	0	0
0	0	0	0	0	0	0	0	0	0	0	0	16:p	1	1	0	0	0	1	0	0	0	0	1	0
0	0	0	0	0	0	0	0	0	0	0	0	17:p	1	1	0	0	0	0	1	0	0	0	0	1
0	0	0	0	0	0	0	0	0	0	0	0	18:p	1	1	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	19:p	1	0	1	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	20:p	1	0	0	1	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	21:p	1	0	0	0	1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	22:p	1	0	0	0	0	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	23:p	1	0	0	0	0	0	1	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	24:p	1	0	0	0	0	0	0	0	0	0	0	0

The relevant sections of the full output for this model are shown at the top of the next page. We see clearly that from the conditional **S**-vector that 3 parameters are below the threshold ($24 - 3 = 21$ estimated parameters, as indicated).



So, what does this have to do with how we coded the design matrix? Well, think about what this design matrix does – it makes the final time step the reference. And this might be a problem...because? Because the final time step involves a non-identifiable β term. So, we're using for our reference a term which can't be separately identified anyway! Perhaps this isn't such a good idea.

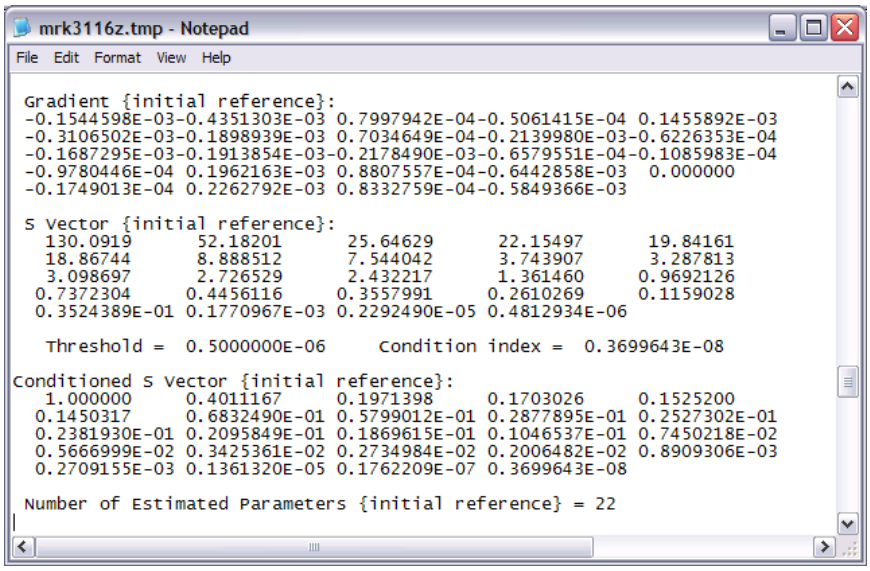
What happens if instead we make the *first* time step the reference. We show this in the following design matrix:



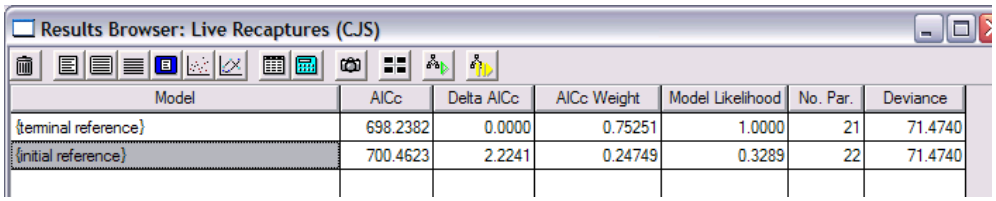
Look closely, and compare it to the default coding (on the preceding page). Make sure you see the differences.

Now, does this subtle change in what we use for the reference change our results? In this case, yes – if you fit a model using this design matrix to the Dipper data, **MARK** will correctly report 22 parameters. The offending parameter $p_{m,3}$ is now estimated and counted properly.

Here is the conditional *S*-vector for this model:



Note that in the results browser (shown at the top of the next page), the two models have exactly the same model deviance – the two models have the identical likelihood and deviance values, but different conditional **S**-vectors – leading **MARK** to conclude they have a different number of estimable parameters, which they clearly should not (since they are equivalent models).



Model	AICc	Delta AICc	AICc Weight	Model Likelihood	No. Par.	Deviance
{terminal reference}	698.2382	0.0000	0.75251	1.0000	21	71.4740
{initial reference}	700.4623	2.2241	0.24749	0.3289	22	71.4740

While we selected this 'pathological' example intentionally, the general point we want to make is that *you probably should not make a non-identifiable parameter the reference cell in your design matrix*. Doing so can have unintended results, as this example shows. As noted earlier (- `sidebar` -, p. 18) you can set a preference in **MARK** to use the first row (i.e., first interval or occasion) as the reference by default.

end sidebar

6.14. A final example: mean values

A final example to emphasize the power and flexibility of design matrices in **MARK**. Suppose you're working on a final report for some study of some organism. In your report, you've been asked to report the 'average' survival value over the years of the study. Now, if a model where apparent survival is constant over time (i.e., $\varphi_{.}$) fits the data much, much better than the model where survival is allowed to vary over time (i.e., φ_t), then it might be reasonable to simply report the constant survival estimate as the mean. However, suppose instead that the model with time-specific variation in survival is strongly supported – what do you do then? Well, the obvious answer might be to simply add up the individual

time-specific estimates, and take the arithmetic average. Is this correct? What about the SE for this average?

In fact, the most direct (and rigorous) way to estimate the mean survival over time, and the appropriate standard error, is to fit a model with an appropriately coded design matrix. To show how this might be accomplished, assume a design matrix with 4 estimates of survival. The default in **MARK** for a time-specific estimate of survival would be a (4×4) identity matrix. Recall that for the identity matrix, each row corresponds to a real parameter, and each column corresponds to a β parameter. Thus, each β parameter represents a treatment estimate (where in this case, each level of the 'treatment' corresponds to a different time interval). Alternatively, we could use the intercept-based 'reference' coding we've used throughout most of this chapter – for this example, with 4 intervals, we would use

$$\mathbf{X} = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

Now recall from earlier in the chapter we saw that there was yet another way we could modify the design matrix:

$$\mathbf{X} = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & -1 & -1 & -1 \end{bmatrix}$$

With this design matrix, the back-transformed β_1 provides the *mean* of the survival estimates. The estimates of β_2 through β_4 provide the time variation around the estimated mean. To see how this works, compute the mean of the rows of the matrix: $[(\beta_1 + \beta_2) + (\beta_1 + \beta_3) + (\beta_1 + \beta_4) + (\beta_1 - \beta_2 - \beta_3 - \beta_4)]/4 = 4\beta_1/4 = \beta_1$. Pretty nifty!

Now, to obtain the estimate of mean survival, the estimate of β_1 must be transformed using the link function used for the model. For example, with the logit link, we write

$$\hat{\bar{S}} = \frac{1}{1 + e^{-\hat{\beta}_1}}$$

So, again we see that by use of the design matrix, we can estimate many things of interest.

begin sidebar

Caution 1: estimating the mean

Now, while the preceding discussion seems fairly straightforward, there is a small little problem involving 'bias'. It turns out that estimates of the mean are potentially biased as a result of transforming back and forth from the non-linear link function. For example, suppose S_i are 0.5, 0.6, 0.7 and 0.8. The arithmetic mean of these 4 values is 0.65. With the logit link, the transformed values are 0, 0.40547, 0.8473 and 1.38629, respectively, giving an intercept of 0.659765. Back transforming from this value gives 0.6592, not 0.6500. Thus, all of the link functions in **MARK**, except the identity link (the only 'linear' link function), will provide slightly biased estimates of the mean value. (Note: the direction of the bias in the logit link will vary depending on whether the mean parameter value is above or below 0.5). Because the identity link is a linear function, estimates of the mean calculated using the identity link will be unbiased; however, the identity link doesn't always work. In most cases, standard errors of the estimates will typically dominate such transformation bias, so that the bias in the back-transformed estimate of the mean is usually ignored.

Caution 2: 'dot' models and other approaches

You might wonder 'why not fit a time-invariant 'dot model' to derive mean values for the vital rates?'. After all, a 'dot model' yields the same value for all intervals.

Unfortunately, this approach, while simple to implement, is not strictly correct. While the estimated 'mean' from the 'dot' model will be relatively robust (i.e., fairly unbiased), the estimated SE will be negatively biased wrt to the true process variance – often by a considerable amount. Here is a simple example – we simulated a data set of live-encounter data, 16 occasions (15 intervals; 250 newly marked individuals released at each occasion), where 'true survival' over a given interval was generated by selecting a random normal deviate drawn from $\mathcal{N}(0.7, 0.005)$ (in the generating model, encounter probability p was constant; $p = 0.35$).

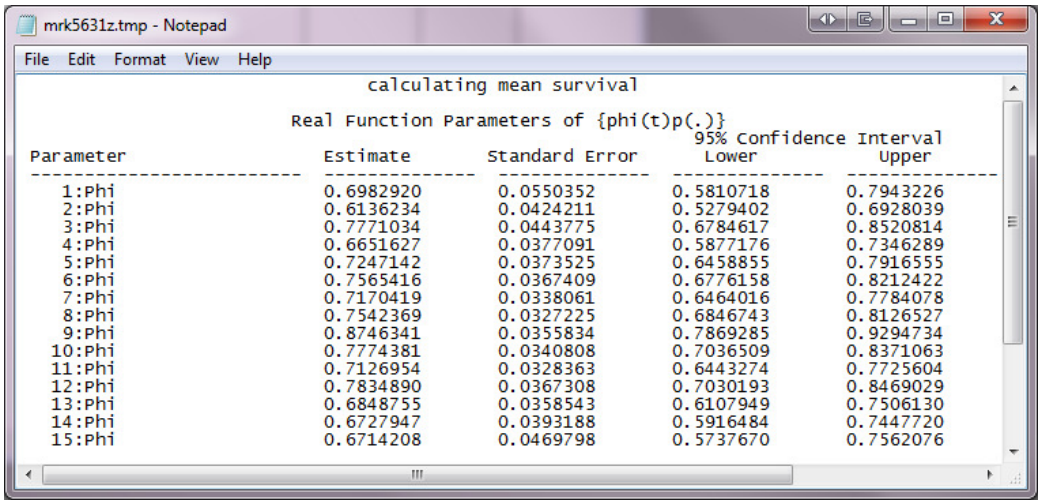
Here is the set (i.e., 'sample' from the underlying random distribution) of the 15 parameter values we used for simulating survival for each of the 15 intervals:

$$\{0.66, 0.62, 0.78, 0.69, 0.72, 0.71, 0.80, 0.76, 0.87, 0.72, 0.74, 0.76, 0.70, 0.68, 0.59\}.$$

Based on these values, the true *sample* mean survival is $\bar{S} = 0.72$, with a true *sample* process variance of $\sigma^2 = 0.005$ (convenient that it matches the variance of the distribution used to generate the values in the first place).

We'll consider 3 different approaches to estimating the mean survival probability: (i) using the estimate of $\hat{\varphi}$ from a 'dot' model, $\{S_{\cdot}p_{\cdot}\}$, (ii) using the estimates from a design matrix for a $\{S_t p_{\cdot}\}$ model constructed such that one of the estimated β terms (the intercept) corresponds directly to the mean, and (iii) a variance components decomposition (for the purposes of demonstrating the most appropriate approach – the subject of random effects models and variance components analysis is considered in some detail in Appendix D).

Fitting the true model $\{S_t p_{\cdot}\}$ to the data yielded the following estimates of survival:


mrk5631z.tmp - Notepad

File Edit Format View Help

calculating mean survival

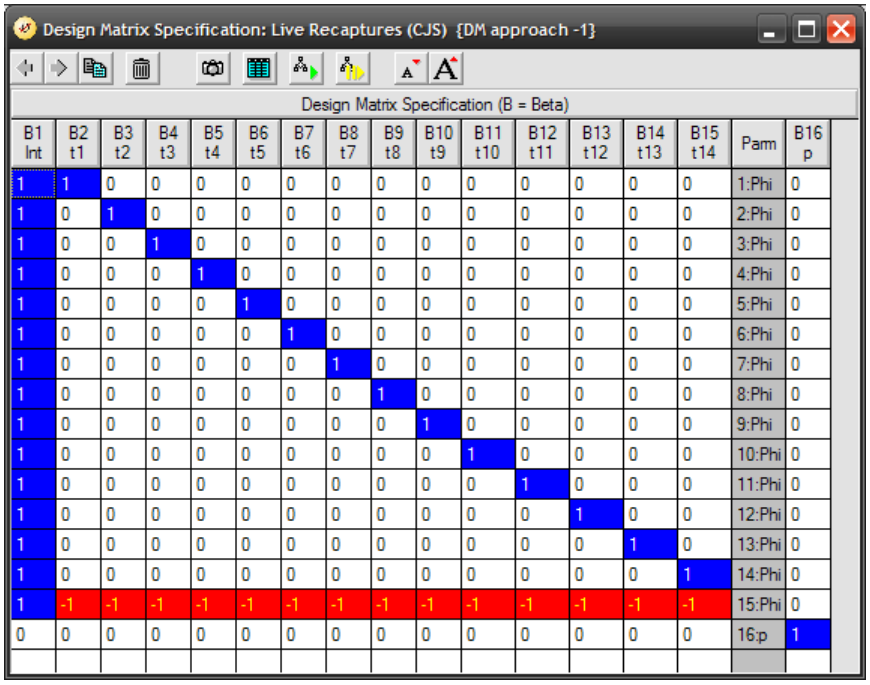
Real Function Parameters of {phi(t)p(.)}

Parameter	Estimate	Standard Error	95% Confidence Interval Lower	Upper
1:Phi	0.6982920	0.0550352	0.5810718	0.7943226
2:Phi	0.6136234	0.0424211	0.5279402	0.6928039
3:Phi	0.7771034	0.0443775	0.6784617	0.8520814
4:Phi	0.6651627	0.0377091	0.5877176	0.7346289
5:Phi	0.7247142	0.0373525	0.6458855	0.7916555
6:Phi	0.7565416	0.0367409	0.6776158	0.8212422
7:Phi	0.7170419	0.0338061	0.6464016	0.7784078
8:Phi	0.7542369	0.0327225	0.6846743	0.8126527
9:Phi	0.8746341	0.0355834	0.7869285	0.9294734
10:Phi	0.7774381	0.0340808	0.7036509	0.8371063
11:Phi	0.7126954	0.0328363	0.6443274	0.7725604
12:Phi	0.7834890	0.0367308	0.7030193	0.8469029
13:Phi	0.6848755	0.0358543	0.6107949	0.7506130
14:Phi	0.6727947	0.0393188	0.5916484	0.7447720
15:Phi	0.6714208	0.0469798	0.5737670	0.7562076

The simple arithmetic mean from these estimates was $\hat{\bar{S}} = 0.7205$, quite close to the true sample mean of $\bar{S} = 0.72$. The naive estimate for the variance is 0.00878, somewhat higher than the true underlying process variation $\sigma^2 = 0.005$. As discussed in Appendix D, this is because the naive estimate includes sampling as well as process variation (and thus will generally be larger than process variation alone).

If we fit the data using model $\{S_{\cdot}p_{\cdot}\}$, the estimate for survival was $\hat{\bar{S}} = 0.7411$, while the variance of the estimates was $\hat{\sigma}^2 = 0.000075$. Clearly, the estimate for survival is relatively close to the true value of the mean (perhaps within comfort limits), but the estimated variance is now very strongly negatively biased. So, the 'dot' model does reasonably well at estimating the mean, but fails miserably at estimating the correct precision of the mean.

Next, we try fitting the data using the PIM for the true model $\{S_t p_\cdot\}$, using a design matrix constructed such that one of the estimated β terms (the intercept) corresponds directly to the mean (shown below):

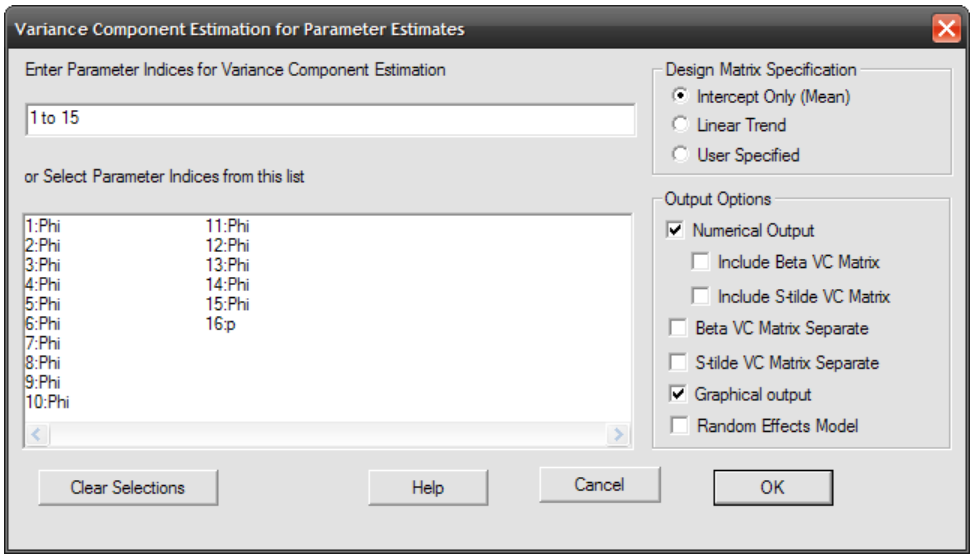


B1 Int	B2 t1	B3 t2	B4 t3	B5 t4	B6 t5	B7 t6	B8 t7	B9 t8	B10 t9	B11 t10	B12 t11	B13 t12	B14 t13	B15 t14	Parm	B16 p
1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1:Phi	0
1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	2:Phi	0
1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	3:Phi	0
1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	4:Phi	0
1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	5:Phi	0
1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	6:Phi	0
1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	7:Phi	0
1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	8:Phi	0
1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	9:Phi	0
1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	10:Phi	0
1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	11:Phi	0
1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	12:Phi	0
1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	13:Phi	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	14:Phi	0
1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	15:Phi	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	16:p	1

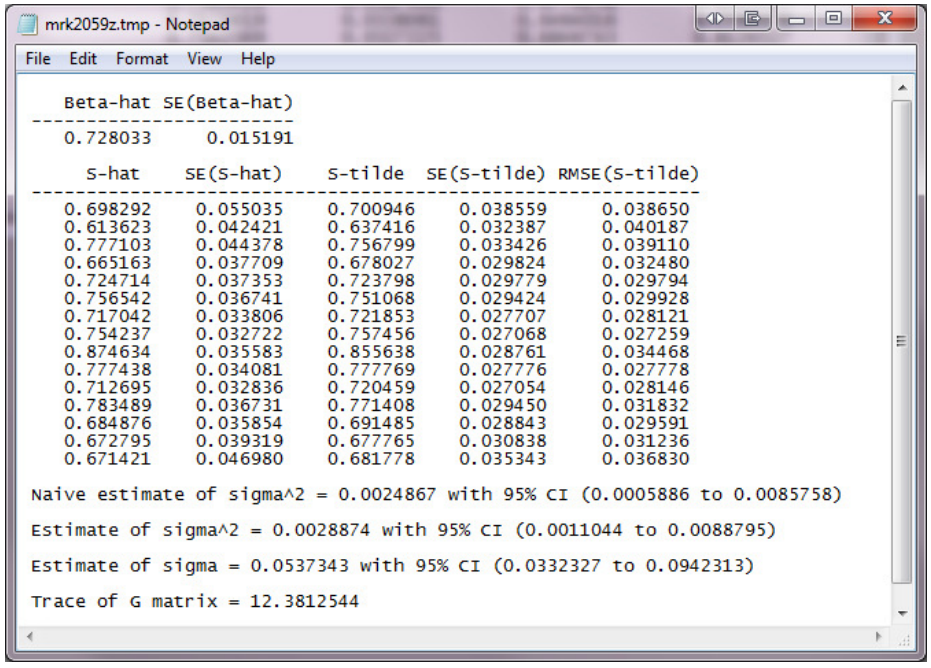
When run using an identity link, the estimate for the first β term in this design matrix (β_1) is the estimated mean survival probability. For our simulated data, the estimated mean was $\hat{\beta}_1 = 0.7205$, which is *identical* to the arithmetic mean of the individual estimates derived from model $\{\varphi_t p_\cdot\}$. However, the estimated variance was $\hat{\sigma}^2 = 0.000099$ – again, much too small.

The reason for the difference is that the variance estimates from both the 'dot' model and DM approach include only the sampling variation, and do not include the process variance associated with the set of estimates (the 'dot' model is the smaller of the two since in effect it represents a fixed effects design with only one replicate, whereas the DM approach has $k - 1$ replicates). The sampling variance is expected to be quite small (relative to process variance) because the sample size is relatively large for this example.

Finally, we apply a variance components approach (based on a random effects intercept only model). While variance components and random effects models is presented in more detail in Appendix D, for now we'll demonstrate the simple mechanics of deriving an estimate of the mean, and process variation using the variance components capabilities in **MARK**. In fact, for the simple objective at hand, the mechanics are very easy. First, retrieve model $\{\varphi_t p_\cdot\}$ in the browser. Then, select '**`Output | Specific Model Output | Variance Components | Real Parameter Estimates`**'. This will spawn another window (shown at the top of the next page) asking you to select the parameter(s) you want to work with, plus some other options. On the right hand side, we've selected '**`intercept only`**' (corresponding to the mean only model), plus various output options. We'll defer 'deep understanding' of what the various options in this window 'do' for now.



Once you hit the '**OK**' button, **MARK** will dump various 'estimates' into the editor. The estimates for our example data are shown below:



The top line gives the estimated mean ('Beta-hat') as 0.7266, which is close to (but slightly different than) the simple arithmetic mean. Both are very close to the true mean of 0.72. More important, here, is the reported SE(Beta-hat) of 0.019640. SE(Beta-hat) is the standard error of the estimated mean and includes *both* process *and* sampling variance.

The table immediately below these two values represent the interval-specific estimates, their standard errors, and corresponding shrinkage estimates (if you don't know what a 'shrinkage' estimate is, don't worry at this stage). Skipping the technical details, these shrinkage estimates are part of

the calculation(s) **MARK** uses to separate the process and sampling variance. This is followed by two 'estimates' of the *process* variance (i.e., the total variance minus the sampling variance) – the `'naive estimate of sigma^2'` (which is the average of the total variance minus the average of the sampling variances), and the `'estimate of sigma^2'` (from Burnham's 'moments' estimator). These are the estimates we're after – for various technical reasons (see Appendix D), the second estimate is preferred. For our simulated data, the process variance is calculated as 0.0045, which is very close to the true value 0.005.

Note that the reported values for our 'variance components' analysis of our simulated data were very close to the true values, because we applied the decomposition to the true generating model. If we had used a different model, we would generate different estimates of the process variance. In fact, in general, you would normally want to estimate the mean of a parameter with full time-dependence in the other structural parameters (i.e., model $\{\varphi_t p_t\}$ in this case). We used the reduced parameter model here only because it was the true generating model under which the data were simulated.

In conclusion, to get a robust estimate of the process variance, and a less-biased estimate of the mean, you'd need to use a random effects (variance components) approach.

end sidebar

6.15. RMark – an alternative approach to linear models

If you've made it this far, then you probably have a pretty good feel for the relationship between the design matrix and linear models in **MARK**. Good! But, by now, you may have already run into a few instances – even with our relatively simple practice examples – where you've made a typo (or several) in entering the appropriate design matrix. Or, in some cases, it may not be clear how to construct the design matrix corresponding to the linear model of interest. While you will get better with practice, it is also probably true that the 'mechanics' of designing and building design matrices in **MARK** is *the* single greatest source of 'frustration'. This is especially true for very large design matrices, which may include a large number of parameters, and complicated ultrastructural relationships within a parameter.

Recently, Jeff Laake (National Marine Mammal Laboratory) has developed a comprehensive library of functions for the **R** statistical package called **RMark** (naturally), which allow you to build, and analyze, linear models in **MARK** – without ever having to 'get your hands dirty' with design matrices. In effect, what **RMark** does is provide a logically consistent 'natural language' (well, the **R** scripting language is becoming sufficiently familiar that it is relatively 'natural') way of building models – analogous to what you might do in **SAS** (or, in the current context, **M-SURGE**). **RMark** is a very robust, and elegant way to build a large number of complex models, quickly, and relatively easily. There is a fair learning curve (somewhat conditional on how much prior experience you may have with **R**, if any), but once you've mastered the conceptual material presented in this book, it is well worth exploring **RMark** – details and full documentation are provided in Appendix C.

6.16. Summary

We're done...at last! Chapter 6 is a **long** chapter, with many concepts and technical details. However, it is also one of the most important chapters, since it covers one of the most useful tools available with **MARK**. It is **very** important that you understand this material, so if you're at all unsure of the material, go through it again. Your efforts will ultimately be rewarded – you'll find that using the linear models approach with **MARK** will enable you to fit a wide variety of complex analytical models, quickly and easily.

CHAPTER 7

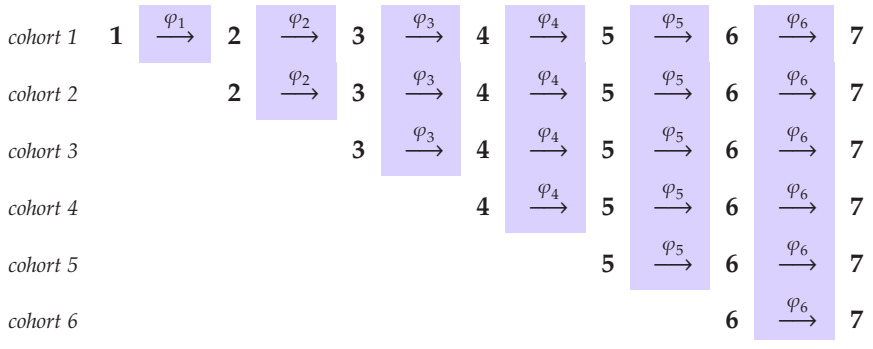
'Age' and cohort models...

Up until now we have assumed that our 'underlying' model has been the CJS time-dependent model. This has been our usual starting point. However, it is obvious to most biologists that there are many instances when the CJS 'assumptions' are not met. By 'assumptions' we are referring to the assumptions concerning independence of fate and identity of rates among individuals (the *iii* assumptions - see Lebreton *et al*. 1992). More specifically, the CJS model generally assumes that 'all individuals, whatever their age or capture history, should have the same probabilities of capture and survival' (Crock 1979 – cited in Lebreton *et al*. 1992).

Clearly, we do not expect this to be generally true. For example, we know that all populations are characterized by differences among individuals. In some cases, such differences are related to one or more particular demographic variables that are 'permanent' (i.e., that don't change over the course of the lifetime of the individual), but which may influence survival, encounter probability, or both.

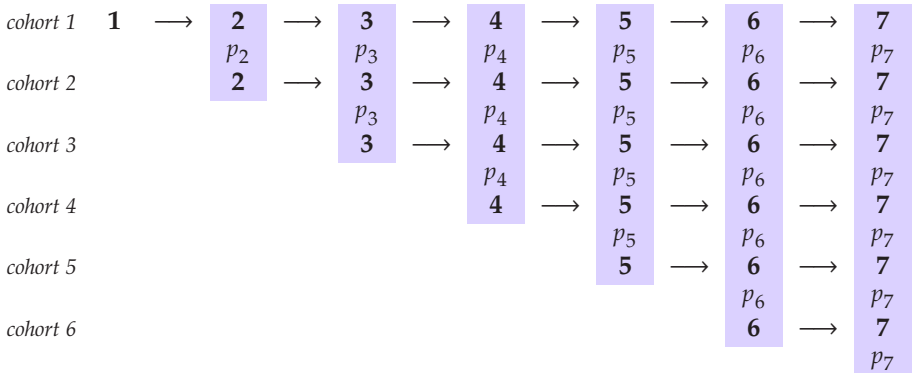
In this chapter, we consider models which account for two common sources of differences among individuals: one that changes over the course of an individuals life (**age**), and one that does not (**cohort**). In both cases, we anticipate that differences in the age or cohort of an individual may influence its survival or encounter probability. For example, we might have strong reason to suspect that the survival of a young individual may generally be lower than the survival of an older individual.

We begin by reviewing the standard CJS model structure for a simple live encounter study. Recall the basic structure of the CJS time-dependent model – shown below for apparent survival φ_i:



© Cooch & White (2014)

and for recapture (encounter) probability, p_i



Under the 'standard assumptions', we assume that survival and recapture potentially vary only as a function of the time interval. However, clearly this might not always be the case – both the time interval *relative* to when the individual was marked, and the time at which the individual was marked, may also be important. We start with the former – 'time since marking' (TSM).

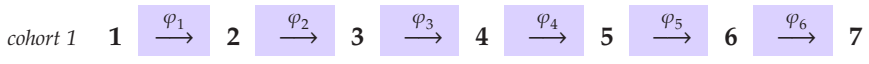
7.1. 'Age' models

It is perhaps exceedingly obvious to most biologists that individuals of different age classes (or developmental stages) differ in the probability of surviving to the next age or stage. In fact, life history theory is to a large degree focussed on analysis of such differences. The reasons for this are well-established. Organisms of a given age (for simplicity, we will refer only to age transitions – the logic applies reasonably well to simple stage-structured systems as well) may be more or less vulnerable to sources of mortality than are individuals of different age(s). The reasons for these differences might reflect differences in size, behavior, or physiological maturation. It is probably safe to say that there have been as many papers in the ecological and evolutionary literature devoted to 'age-dependence' of one trait or another as any other subject. So, we need to be able to address the question: are there age-specific differences in survival? To do that, we need to consider the construction of 'age' models (the reason for referring to 'age' parenthetically will become clear later in the chapter).

First, some definitions. Clearly, the ageing process begins when individuals are born. All individuals born in a given breeding season can be said to belong to the same birth cohort. A cohort is simply some criterion by which individuals are grouped together (birth year, in this case). Within a birth cohort, age and time are logically synonymous. In fact, age, time (e.g., year) and cohort are related by the following expression:

age = current year - birth cohort

Consider the first row extracted from the triangular matrix (PIM) of a simple CJS time-dependence survival model.



The first row also corresponds to the first cohort. Let's assume that this is a 'birth' cohort. All individuals that are newly marked and released at occasion 1 were marked as newborns. Let's also

assume (for simplicity) that the sampling occasions correspond to 'years'. Thus, individuals marked at occasion 1 (0 years of age) are, if they survive, 1 year of age at occasion 2, 2 years of age at occasion 3, and so forth.

Now, the parameters shown in this table ($\varphi_1 \rightarrow \varphi_6$) were originally written to show simple time-dependence. But, since individuals also age through time, we cannot differentiate between age-specific differences in survival, and simple time-specific differences. They are completely analogous.

How do we then separate age and time effects? Clearly, this cannot be accomplished using a single cohort alone. However, what happens if we use multiple birth-cohorts? To examine this situation, let's make the following assumptions about some arbitrary population. First, let's assume that all individuals in each cohort are marked as newborns. Let's also assume that survival between age 0 and age 1 year is different from survival after age 1 year. For simplicity, let's call the survival from $0 \rightarrow 1$ 'juvenile' survival, and survival from any age x (where $x \geq 1$ year) to $x+1$ 'adult' survival.

If you think about it, this is not at all an uncommon situation. However, let's make it even simpler. Let's assume for the time being that juvenile survival is constant among cohorts, and that adult survival is constant both within cohorts and among cohorts. What would the parameter structure of this model look like? Let's use the 'j' subscript for juvenile survival, and the 'a' subscript for adult survival.

cohort 1	1	$\xrightarrow{\varphi_j}$	2	$\xrightarrow{\varphi_a}$	3	$\xrightarrow{\varphi_a}$	4	$\xrightarrow{\varphi_a}$	5
cohort 2			2	$\xrightarrow{\varphi_j}$	3	$\xrightarrow{\varphi_a}$	4	$\xrightarrow{\varphi_a}$	5
cohort 3					3	$\xrightarrow{\varphi_j}$	4	$\xrightarrow{\varphi_a}$	5
`cohort 4`							4	$\xrightarrow{\varphi_j}$	5

Now, let's look at this table and see how it makes sense. The best starting point is to look at each cohort separately. Within a cohort, we see that between the first occasion (in the cohort), and the second occasion after marking, individuals survive with probability φ_j. However, from the second occasion after marking onwards (within a cohort), they survive with probability φ_a.

Now, as we discussed previously, within a cohort we cannot differentiate between 'time' and 'age'. However, note that we can test whether a model with this structure differs from one where (say) survival is constant (no age or time effect). But what we're really after is 'age' as separate from 'time'. Here is where multiple cohorts come in. By contrasting parameter estimates among cohorts, but within a time interval, we can differentiate age and time effects. For example, concentrate on the interval between occasion 2 and occasion 3 (shaded area - below).

cohort 1	1	$\xrightarrow{\varphi_j}$	2	$\xrightarrow{\varphi_a}$	3	$\xrightarrow{\varphi_a}$	4	$\xrightarrow{\varphi_a}$	5
cohort 2			2	$\xrightarrow{\varphi_j}$	3	$\xrightarrow{\varphi_a}$	4	$\xrightarrow{\varphi_a}$	5
cohort 3					3	$\xrightarrow{\varphi_j}$	4	$\xrightarrow{\varphi_a}$	5
cohort 4							4	$\xrightarrow{\varphi_j}$	5

In terms of time, both cohorts 1 and 2 are experiencing the same 'temporal effect'. In other words, all individuals, whether they were newly marked at occasion 1 or occasion 2, are experiencing the aspects or characteristics of the 'environment' causing mortality during this interval. However, in cohort 1, the individuals at occasion 2 are 1 year of age, while those from cohort 2 are newborn. Thus, for cohort 1

individuals, they will survive from 2 → 3 with probability φ_a. In contrast, for cohort 2 individuals, they are surviving from 2 → 3 with probability φ_j.

If there were no age-specific differences in survival, then the ratio of survival of cohort 1 individuals over this interval would be the same as the survival of cohort 2 individuals over this interval. In other words, the ratio of φ_a/φ_j would equal 1. Again, it is the contrast *among* cohorts (i.e., rows), but *within* columns (i.e., intervals between occasions) that allows us to test for age differences in survival. This is a very important concept to grasp, so it is critical that you spend the time now to make sure you do.

Let's start to formalize our notation a bit. First, consider the question 'what is the time axis of our model?'. The time axis which we need to follow in an age-structured model is along the diagonal. For example, consider again our model with constant juvenile and constant adult survival, we essentially have 2 parameters. The differently shaded areas of our model (below) show the juvenile (along the diagonal) and adult age classes (above the diagonal), respectively:

cohort 1	1	φ_j →	2	φ_a →	3	φ_a →	4	φ_a → 5
cohort 2			2	φ_j →	3	φ_a →	4	φ_a → 5
cohort 3					3	φ_j →	4	φ_a → 5
cohort 4							4	φ_j → 5

The juvenile age class in this example spans one time interval (i.e., one year). The adult age class (above the diagonal) spans $n - 1$ years, where n is the number of occasions, and '1' is the duration of the juvenile age class. If we re-write the matrix using numbers for subscripts instead of letters (let '1' = '*j*', and '2' = '*a*'), then our model with constant juvenile and adult survival would be

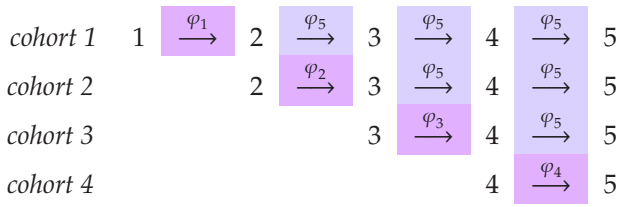
cohort 1	1	φ_1 →	2	φ_2 →	3	φ_2 →	4	φ_2 → 5
cohort 2			2	φ_1 →	3	φ_2 →	4	φ_2 → 5
cohort 3					3	φ_1 →	4	φ_2 → 5
cohort 4							4	φ_1 → 5

Now, if we simplify this and re-write the parameter structure the way that **MARK** interprets it (using only the subscripts), this 2 age-class model (juvenile, adult) is written as:

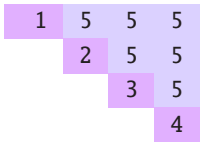
1	2	2	2
	1	2	2
		1	2
			1

OK, now let's expand our model somewhat, adding some more 'flexibility'. Suppose for example that juvenile survival varies over time, but that adult survival is constant through time.

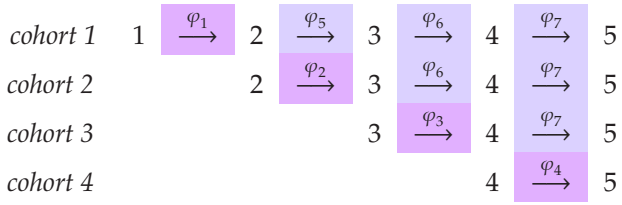
If we now add time-dependence to the juvenile survival probability, but leave adult survival constant, the model structure would now look like:



In the **MARK** parameter (PIM) format, this would reduce to:

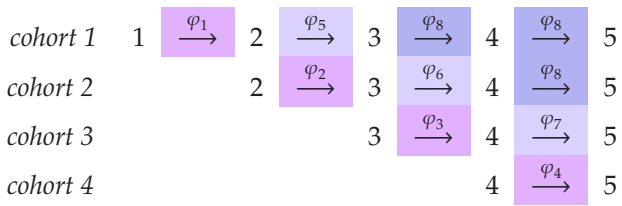


Let's extend it one more step: let's add time dependence to the adult survival as well. This particular model is important since 2 age-class models, with a juvenile age-class spanning one year, and a single adult age class, with time-dependence in both, are very commonly seen in analysis of mark-recapture data from wild populations. Here is what the 2 age-class model with time-dependence in both age classes would look like:



As a final test, to see if you really understand the structure of these models, consider the following situation. We have a 5 occasion study of a long-lived organism with indeterminate growth, and we believe that survival may be age-dependent. We decide to model survival in the following way. 3 age classes, the first age class spanning 1 year, the second age class spanning 1 year, and the final age class spanning all remaining years. In other words, a single-year duration 'juvenile' phase, a 1-year duration 'sub-adult' phase, and the final 'adult' phase. Juvenile and sub-adult survival are time-dependent, but adult survival is constant over time.

Here is the structure for this model:



With a bit of thought, you should be able to see how this model was constructed (look carefully at the ordering of the parameter subscripts). If not, go back through the preceding few pages, and try again. Age models are very important.

If you do 'get the basic idea', then let's proceed to analysis of some simple 2 age-class models. We will examine how to modify the standard CJS PIMs different for types of age models. Note however that modifying the PIMs will let you construct an age (or cohort) model of arbitrary design. We will use the sample data set `AGE.INP`. We 'suspect' that there are 2 age-classes in these data. We want to 'confirm' our suspicion by using **MARK** to test the fit of various 2 age-class models versus the standard time-dependent CJS model with no age effects. There is only one group in the data set, and 7 occasions (1 marking occasion and 6 recapture occasions). Let the following be our set of candidate models (i.e., the list of those models that we believe, based on our biological expectations, might be appropriate to these data). Note the subscripting conventions - with more 'structure' in our models (i.e., time, age, cohort), the subscripting can get a bit tricky to handle in any intelligible fashion.

model	*description*
$\varphi_t p_t$	standard CJS model - no 'age structure' - time dependence in both survival and recapture
$\varphi_{a2-./.} p_t$	2 age-classes for survival (`a2`) - both age classes constant (`./.`) through time. Time-dependent recapture.
$\varphi_{a2-t/t} p_t$	2 age-classes for survival (`a2`) - both age classes time-dependent (`t/t`) through time. Time-dependent recapture.
$\varphi_{a2-t/.} p_t$	2 age-classes for survival (`a2`) - juvenile (first) age classes time-dependent, adult age class constant over time (`t/.`) through time. Time-dependent recapture.

Clearly, we are focussing only on the survival side of things with these 4 models. However, do not get the idea that age-effects are only relevant to survival. They're not, and are equally as likely to show up in recapture probabilities as well. As a general 'rule' - do not overlook modeling the recapture side of things with as much interest and care as you do the survival side. We often tend to focus on the 'finality' of a survival analysis (since death has rather obvious fitness consequences), but why an individual isn't seen on a given occasion can be of equal interest.

By now, you should be able to start **MARK** and run the standard CJS model almost by reflex - go ahead and do so, and add the results to the browser. Clearly, the CJS model makes a reasonable 'null' to test against - it is well-parameterized, but has no 'age structure'. Once the CJS model run is complete, we'll proceed and run the other 3 candidate models, in order, starting with $\{\varphi_{a2-./.} p_t\}$ - 2 age-classes for survival, constant over time for both ages, and time-dependent recapture probability. How do we run this in **MARK**? By now, you should recognize that virtually *all* models in **MARK** are set by manipulating the PIMs and/or the design matrix, either alone or in combination (if you don't realize this, for shame – and go back and re-read Chapter 6). In this case, since all we're doing is changing the way in which the parameters are indexed (to reflect the age-structure), we modify the PIMs. So, bring up the survival and recapture PIMs.

Obviously, the recapture PIM will 'look' like the standard CJS PIM we've seen many times already – full time-dependence (this is what **MARK** defaults to). But what about the survival PIM - how do we modify to reflect the model structure $\{\varphi_{a2-./.} p_t\}$? Actually, for this particular model, it's pretty easy –

MARK makes this model one of its built-in menu options. As a first step, you should be able to write out what the PIM should look like before you construct it.

In this case, with 7 occasions, the PIM should reflect the following structure:

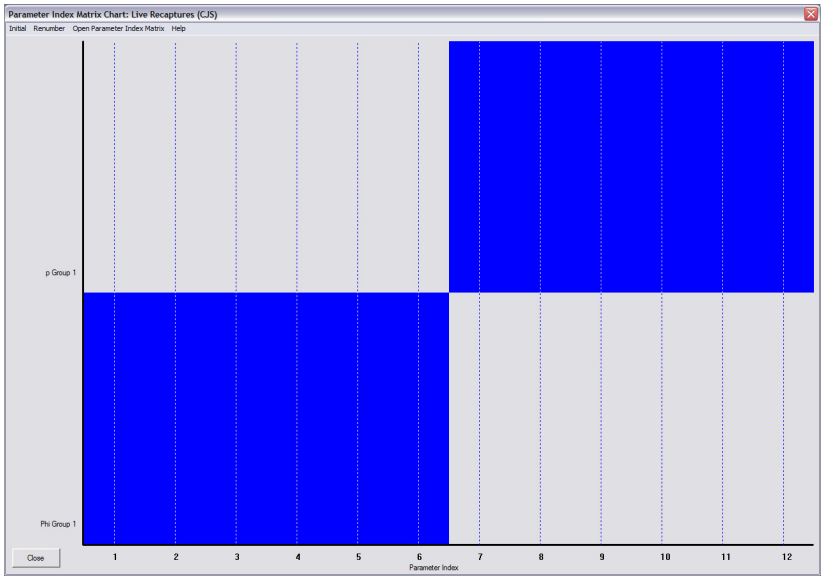
cohort 1	1	φ_1	2	φ_2	3	φ_2	4	φ_2	5	φ_2	6	φ_2	7
cohort 2			2	φ_1	3	φ_2	4	φ_2	5	φ_2	6	φ_2	7
cohort 3					3	φ_1	4	φ_2	5	φ_2	6	φ_2	7
cohort 4							4	φ_1	5	φ_2	6	φ_2	7
cohort 5									5	φ_1	6	φ_2	7
cohort 6											6	φ_1	7

which corresponds to...

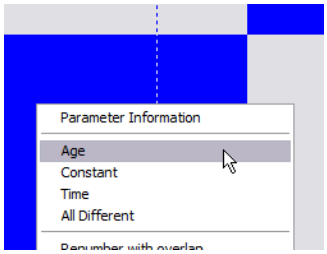
1	2	2	2	2	2
	1	2	2	2	2
		1	2	2	2
			1	2	2
				1	2
					1

Make sure you understand the connection. Pay particular attention to the parameter subscripting (which of course leads to the parameter indexing you need to keep track of when constructing the PIMs, and reading the output).

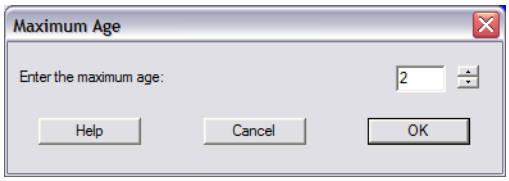
Again, while it would be relatively straightforward to modify the survival PIM to reflect this structure by manually editing each cell, **MARK** makes it much easier for you than that. As we've seen in earlier chapters, one of the handy features of **MARK** is the ability to modify the PIMs through the Parameter Index Chart. Open up the PIM chart, which initially reflects the CJS time-dependent model.



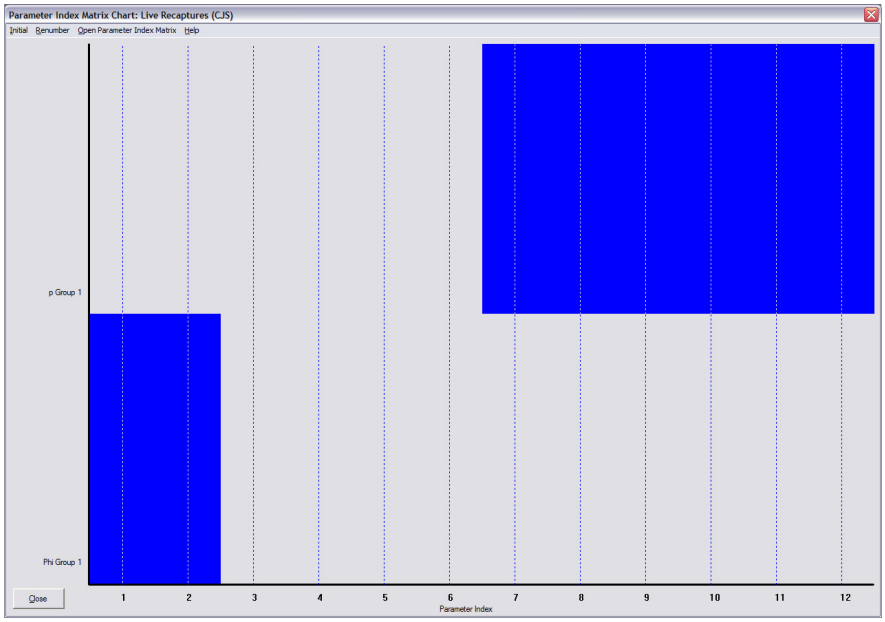
Moving the cursor over the survival 'blue box' in the PIM chart (the one in the lower left-hand corner), right click the mouse. This will spawn a menu allowing you to select from a variety of 'built-in' parameter structures. Obviously, the '**Age**' model is the one we're interested in here, so go ahead and select '**Age**' from this menu.



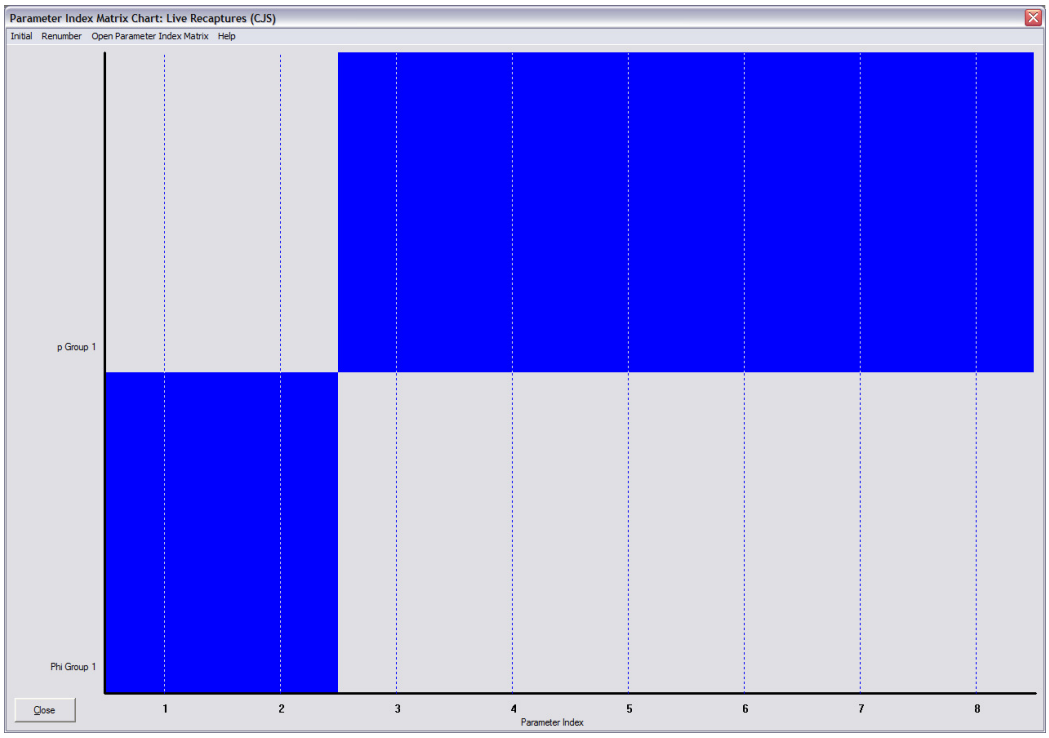
You will then be presented with a small dialogue window asking you to define the age of the oldest age class. Now, this is potentially confusing. If each age class spans one year, this is the same as asking 'how many age classes do you want'? For example, if the oldest age class is 3 years, then there will be 3 age classes, each spanning one year in length. Seems reasonable enough. In our analysis, we want 2 age classes: one for 'juveniles' or 'young' (i.e., the first year after marking as offspring), and 'adult'. So, the oldest age class is 2 years.



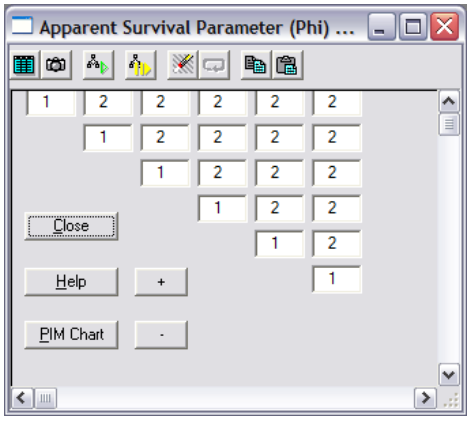
Once you've clicked '**OK**', the PIM chart will reflect this change. It should now look like



To eliminate the 'gap' between the two parameters in the PIM chart, you can simply drag the recapture box over to the left. The PIM chart should now look like

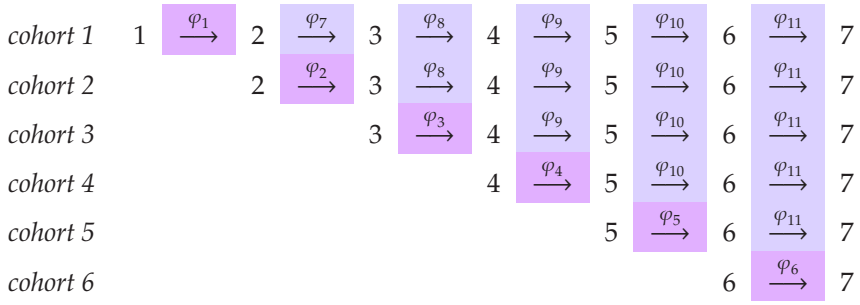


Now, in some senses, the PIM chart doesn't give you the best indication of the actual PIM structure. To look at it (and to confirm the PIM is structured correctly), right-click again over the survival 'box' in the PIM chart, and open the corresponding PIM window (one of the menu options). Then, close the PIM chart and have a look at the survival PIM.

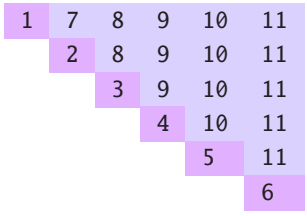


As you can see, it looks exactly as we wanted it to (check back earlier in this chapter if you don't remember the details for this model). Go ahead and run this model - we'll use the sin link, and name the model '`phi(a2-./.)p(t)`'. Add the results to the browser.

The next model is $\{\varphi_{a2-t/t}p_t\}$ - 2 age-classes, but this time with full time-dependence within each age-class. Time-dependent recapture. Here is the structure of this model (and the corresponding PIM):



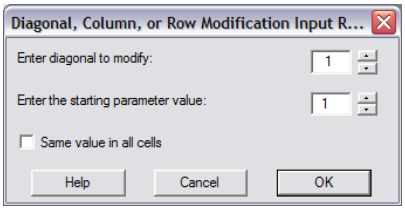
which corresponds to...



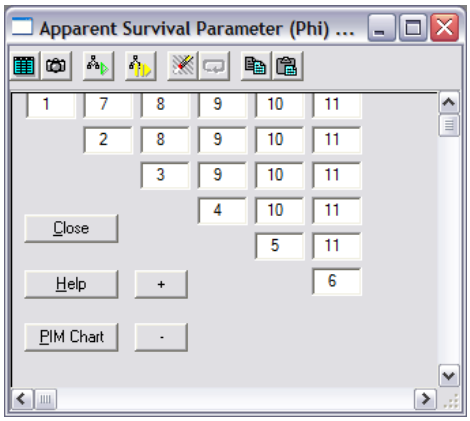
How do you get the PIM to look like this in **MARK**? There is no simple menu option for this model (the menu we used for the previous model only allows you to build age models which are constant over time in each age class). However, here is a relatively simple way to construct the PIM. First, you need to pay attention to the parameter indexing. Note that the largest value on the diagonal representing the first age-class is '6', and the first adult parameter value is (therefore) '7'.

So, here's what you do:

1. make the first cell value '6', instead of '1'
2. pull down the '**Initial**' menu, and select '**Time**'. This will create a PIM that is a standard CJS time-dependent PIM, but one that starts with '6'.
3. next, go back to the first cell, and click it again. Pull down the '**Initial**' menu, and select '**Diagonal**'. When you do, **MARK** will present you with a window asking you which diagonal you want to modify - since you've clicked in the first cell, it will default to the first diagonal, but you can always change this. It will also ask you what you want the starting parameter value to be. Change it to '1', and click '**OK**'.

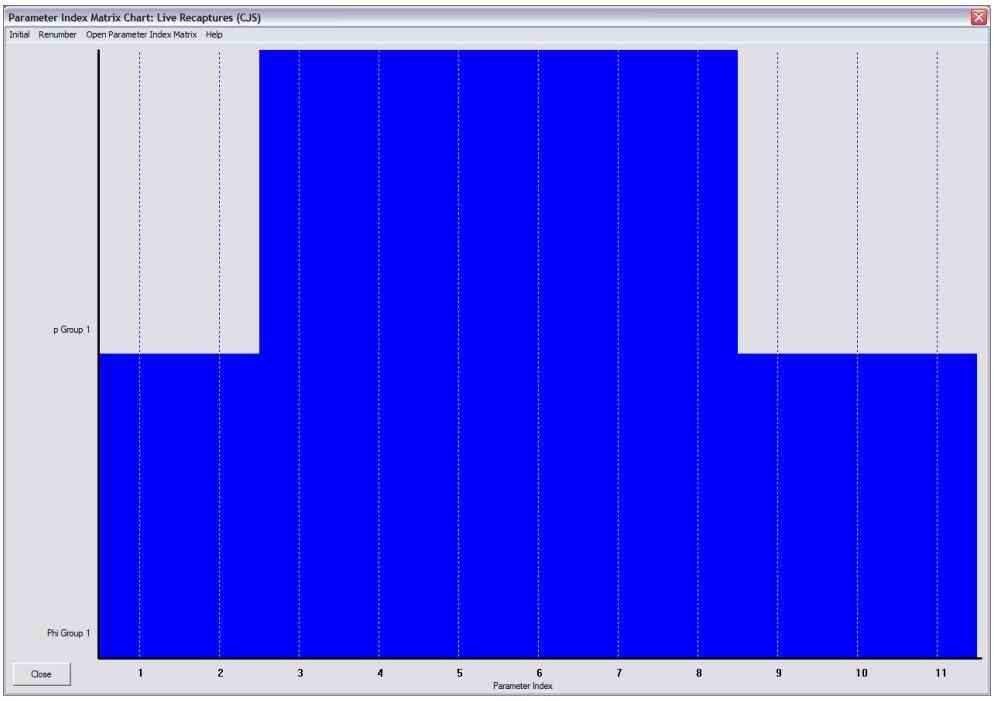


Now, have a look at the PIM - it should look exactly like we want it to - time-dependent indexing along the first diagonal from 1 → 6, and then time-dependence for the second 'adult' age class, from 7 → 11.



What we just did makes use of the fact that this model is basically a modification of the CJS model - in fact, the classic time-dependent structure within cohort is maintained for the second age-class, so all you do is figure out what the first parameter value would need to be for this age class, and go from there. With a bit of thought, we think you'll get the hang of it.

Now, before you run this model - have a look at the relative indexing of the survival and recapture PIMs. You can do this easily by using the Parameter Index Chart, from the '**PIM**' menu. What do we see?



We see that the parameter indexing is 'overlapped' between the 2 parameters - this is **not** good, and is not something that **MARK** will 'fix behind the scenes'. As such, you **must** correct this problem yourself.

The easiest way to do this is by modifying the 'overlap' via the PIM chart itself. You can either (i) drag the recapture 'box' to the right, or (ii) use the '**Renumber without overlap**' option. Simply right-click anywhere in the PIM chart, and select '**Renumber no overlap**'. This will do the trick. But double-check your indexing – we can't stress this enough. The PIM chart can be very useful for doing this. If your parameters are overlapped (as above), **MARK** might run successfully (i.e., it won't crash, burn, fold and mutilate your computer), but your estimates won't reflect the model you're really after. There are cases where overlap is desired, but not here. Once you've corrected the indexing, go ahead and run the model - name it '`Phi(a2-t/t),p(t)`', and add the results to the browser.

The final model is $\{\varphi_{a2-t/.}p_t\}$ - 2 age-classes, with time-dependence in the first age-class, but constant survival in the second age-class. This model is very common for analysis of individuals marked as young - temporal variation in the first age class, but no variation among 'adults'. The idea, of course, is that individuals may be more susceptible to annual variation in conditions affecting survival in their first year than they are if they survive to adulthood. Here is the structure and PIM for this model:

cohort 1	1	φ_1	2	φ_7	3	φ_7	4	φ_7	5	φ_7	6	φ_7	7
cohort 2			2	φ_2	3	φ_7	4	φ_7	5	φ_7	6	φ_7	7
cohort 3					3	φ_3	4	φ_7	5	φ_7	6	φ_7	7
cohort 4							4	φ_4	5	φ_7	6	φ_7	7
cohort 5									5	φ_5	6	φ_7	7
cohort 6											6	φ_6	7

which corresponds to...

1	7	7	7	7	7
	2	7	7	7	7
		3	7	7	7
			4	7	7
				5	7
					6

We'll leave it to you to figure out how to construct this PIM in **MARK** - all you need to do is modify the logic used for the last PIM a bit. Again, check your parameter indexing. Since the largest value in the survival PIM is 7, the first value in the recapture PIM must be at least 8. Go ahead and run the model, name it '`Phi(a2-t/c),p(t)`', and add the results to the browser. There should now be 4 model results in the browser.

Results Browser: Live Recaptures (CJS)

Model	AICc	Delta AICc	AICc Weight	Model Likelihood	No. Par.	Deviance
{phi(a2 - t/.)p(t)}	5053.8746	0.0000	0.67789	1.0000	13	133.0730
{phi(a2 - t/t)p(t)}	5055.3628	1.4882	0.32211	0.4752	16	128.5015
{phi(a2 - ./.)p(t)}	5175.4319	121.5573	0.00000	0.0000	8	264.7031
{phi(t)p(t)}	5301.6450	247.7704	0.00000	0.0000	11	384.8765

We see from the results browser that the CJS model is clearly inappropriate for these data - any of the models with age-structure fit better, by any criterion you want to use (no pun intended). In fact, you might want to run these data through **RELEASE** - by looking carefully at the results from TEST 3, you should see some clue as to the 'sources of lack of fit to the CJS model' - in this case, the source being significant age structure. (By the way, you might be – or should be – asking yourselves about doing a GOF test for the general model, to derive an estimate of $\hat{c}$. The general model in this case is model $\{\varphi_{a2-t/t}p_t\}$. While we could use the bootstrap or median-$\hat{c}$ here, there is no need, since the data are simulated under this model with $\hat{c} = 1.0$).

Among the age models, the model with time-dependence in the first age class, but constant survival among the 'adults' has the lowest AIC value, and is a little over twice as well supported by the data than the next best model (the model with time-dependence in both age classes). The LRT comparison of these 2 best models shows that the difference in fit is not significant ($\chi^2 = 4.57, df = 3, P > 0.2$). In other words, the addition of time-dependence to the 'adult' class did not significantly improve the fit - thus we select the more parsimonious model with constant survival in this age class.

Now, before we go much further, how did **MARK** count the parameters for these models? Again, although **MARK** does very well in parameter counting, it is always a good idea to know yourself how many potential parameters there are – if for no other reason than to be able to spot discrepancies between the 'potential' and 'actual' number of parameters estimated given the data. Let's consider the model with the most parameters - model $\{\varphi_{a2-t/t}p_t\}$, with time-dependence in both age-classes. How many individually identifiable parameters are in this model? The saturated capture histories, and their associated probability functions, are shown at the top of the next page. Make sure you see how these functions were derived from the survival and recapture matrices.

capture history	*probability*
1111111	$\varphi_1 p_2 \varphi_7 p_3 \varphi_8 p_4 \varphi_9 p_5 \varphi_{10} p_6 \varphi_{11} p_7$
0111111	$\varphi_2 p_3 \varphi_8 p_4 \varphi_9 p_5 \varphi_{10} p_6 \varphi_{11} p_7$
0011111	$\varphi_3 p_4 \varphi_9 p_5 \varphi_{10} p_6 \varphi_{11} p_7$
0001111	$\varphi_4 p_5 \varphi_{10} p_6 \varphi_{11} p_7$
0000111	$\varphi_5 p_6 \varphi_{11} p_7$
0000011	$\varphi_6 p_7$

Now, of course, the next step is to determine which of the parameters in this table are individually identifiable. By now you may have realized that the 'critical' part of this process typically involves looking at the terminal products. In this case, we have 2 different products: $\varphi_{11}p_7$ and $\varphi_6 p_7$. Are these β terms? If you think about it for a moment, you might realize that the answer is 'yes' - since p_7 is not identifiable. Thus, 16 identifiable parameters. This model corresponds to Table 7D in Lebreton *et al*. (1992). They note that for this model, the number of identifiable parameters is given as $(3k - 5)$, where k is the number of occasions. Since $k = 7$ in this example, we see that $(3k - 5) = (21 - 5) = 16$ parameters. Which is exactly what **MARK** gives us in the results browser.

What about the 2 other age models - $\{\varphi_{a2-./.}p_t\}$ and $\{\varphi_{a2-t/.}p_t\}$? Start with the first one - since both adult age classes have constant survival, the saturated capture-histories and their corresponding probability statements are (top of the next page):

capture history	*probability*
1111111	$\varphi_1 p_2 \varphi_2 p_3 \varphi_2 p_4 \varphi_2 p_5 \varphi_2 p_6 \varphi_2 p_7$
0111111	$\varphi_1 p_3 \varphi_2 p_4 \varphi_2 p_5 \varphi_2 p_6 \varphi_2 p_7$
0011111	$\varphi_1 p_4 \varphi_2 p_5 \varphi_2 p_6 \varphi_2 p_7$
0001111	$\varphi_1 p_5 \varphi_2 p_6 \varphi_2 p_7$
0000111	$\varphi_1 p_6 \varphi_2 p_7$
0000011	$\varphi_1 p_7$

Make sure you can derive the the probability statements for this model yourself. How many estimable parameters? **MARK** tells us there are 8 estimable parameters for this model. Is this consistent with the probability statements? There are clearly 2 potential survival parameters (φ_1 and φ_2), and 6 potential recapture parameters (p_2 to p_7) – 8 total potential parameters. Are they all estimable? If you recall from earlier chapters where we deal with models where one or more parameters were held constant, in such cases, there are generally no confounded terminal p and φ since some parameters are estimable because other parameters are held constant across occasions. This is the case here - **all** 8 parameters are estimable, so **MARK** is correct.

What about the other model - model $\{\varphi_{a2-t/.}p_t\}$ - which has both a constant survival (the 'adult' age-class) and time-dependent survival term (the 'juvenile' age class)? Are all terms estimable? Yes! How many? 13 - exactly what **MARK** tells us: 7 survival parameters, and 6 recapture parameters. Why are φ_6 and p_7 separately estimable? Simply because one of the elements (p_7) is estimable from earlier cohorts.

So, we see that **MARK** has correctly calculated the number of estimable parameters for all 3 age models. However, the only way to know if **MARK** is 'correct' is to know how to count the parameters yourself. It can be a somewhat laborious process (admittedly), but it is probably worth the effort.

7.2. Constraining an age model

Constraining an age model is not much more difficult than constraining a model without age structure, but there are a few things you need to keep in mind. We'll start by looking at how we would construct the design matrix to correspond to the PIM for survival for a simple age model, with 2 age classes, and time-dependence in each age class. We'll assume that we have 7 occasions. Recall that the PIM for this model looks like:

1	7	8	9	10	11
	2	8	9	10	11
		3	9	10	11
			4	10	11
				5	11
					6

So, based on the number of indexed parameters in the PIM, we know already that our design matrix for survival would need to have 11 rows and 11 columns. What does the linear model look like? Again, writing out the linear model is often the easiest place to start. In this case we see that over a given time interval, we have, in effect, 2 kinds of individuals: *juveniles* (individuals in their first year after marking), and *adults* (individuals at least 2 years after marking).

Thus, for a given TIME interval, there are 2 groups: juvenile and adult. If we call this group effect AGE, then we can write out our linear model as

$$\begin{aligned}\text{logit}(\hat{\varphi}) &= \texttt{AGE+TIME+AGE.TIME} \\ &= \beta_1 + \beta_2(\texttt{AGE}) + \beta_3(\texttt{T}_1) + \beta_4(\texttt{T}_2) + \beta_5(\texttt{T}_3) + \beta_6(\texttt{T}_4) + \beta_7(\texttt{T}_5) \\ &\quad + \beta_8(\texttt{AGE.T}_2) + \beta_9(\texttt{AGE.T}_3) + \beta_{10}(\texttt{AGE.T}_4) + \beta_{11}(\texttt{AGE.T}_5)\end{aligned}$$

Whoa - wait a minute! How did you get this linear model from the PIM at the top of the page? If you look closely, you'll see that there is no (AGE*TIME1) interaction term. What's going on here?

OK. . .step by step. The first term (β_1) is the intercept. Easy enough. The second term (β_2) corresponds to the AGE effect (where in this case we have 2 age classes, juvenile and adult - think of the different ages as different levels of a grouping factor AGE). Thus, one 'slope', since the 2 levels of the AGE effect require only 1 column of dummy variables.

The next 5 terms ($\beta_3 \rightarrow \beta_7$) correspond to the first 5 levels of TIME. Recall that there are 6 intervals, but that we need only 5 columns to code for these intervals (since if $\texttt{T}_1 = \texttt{T}_2 = \texttt{T}_3 = \texttt{T}_4 = \texttt{T}_5 = 0$, then the time interval must be 6). Thus, only 5 'slopes' coding for the 6 different time intervals.

Now, the potentially 'tricky' part – the interaction terms. If you look carefully – very carefully – at the linear model equation (above) you'll notice that it did **not** include a term for an ($\texttt{AGE.T}_1$) interaction. Why? You need to think a bit carefully here. Remember, we're treating the two age classes as different groups. If you look back at the examples in Chapter 4, you'll notice that, in those cases, the groups being compared (male vs. female, good vs. poor) were 'temporally symmetrical' - i.e., had the same number of sampling occasions for both groups. In our age example, however, this is **not** the case. Look closely at the PIM - we have 6 parameters for the juvenile age class ($1 \rightarrow 6$) but only 5 parameters for the adult age class ($7 \rightarrow 11$).

Why is this? Well, the answer is obvious if you look back at the structure of our model. We obviously don't have an 'adult' estimate over the first interval for the first cohort, since there were no 'known' adults at that point in the study. Thus, in the first time interval, there are no adults, and thus, there is no logical interaction of the AGE and TIME factors in this interval (since there is only one age class!). The first time interval for which there is a potential interaction of AGE and TIME is interval 2 (look at the PIM to confirm this for yourself). If you don't see the connection between the PIM, and the linear model, take some time now to work through everything, to make sure that you do. It's a **very** important concept.

Now, what does the design matrix look like for our age model, with time-dependence in both age classes? Since the linear model is:

$$\begin{aligned}\text{logit}(\hat{\varphi}) &= \beta_1 + \beta_2(\texttt{AGE}) + \beta_3(\texttt{T}_1) + \beta_4(\texttt{T}_2) + \beta_5(\texttt{T}_3) + \beta_6(\texttt{T}_4) + \beta_7(\texttt{T}_5) \\ &\quad + \beta_8(\texttt{AGE.T}_2) + \beta_9(\texttt{AGE.T}_3) + \beta_{10}(\texttt{AGE.T}_4) + \beta_{11}(\texttt{AGE.T}_5)\end{aligned}$$

then the corresponding design matrix is the one shown at the top of the next page. Make sure you see the connection between the linear model, and this design matrix. Note that if we believed that there was additive variation between juveniles and adults, we would simply modify this design matrix, by deleting the columns corresponding to the interaction terms (columns labeled B8 through B11, corresponding to β_8 through β_{11} in the linear model, respectively).

Design Matrix Specification: Live Recaptures (CJS)

Design Matrix Specification (B = Beta)

B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	B11	Parm
1	1	1	0	0	0	0	0	0	0	0	1:Phi
1	1	0	1	0	0	0	1	0	0	0	2:Phi
1	1	0	0	1	0	0	0	1	0	0	3:Phi
1	1	0	0	0	1	0	0	0	1	0	4:Phi
1	1	0	0	0	0	1	0	0	0	1	5:Phi
1	1	0	0	0	0	0	0	0	0	0	6:Phi
1	0	0	1	0	0	0	0	0	0	0	7:Phi
1	0	0	0	1	0	0	0	0	0	0	8:Phi
1	0	0	0	0	1	0	0	0	0	0	9:Phi
1	0	0	0	0	0	1	0	0	0	0	10:Phi
1	0	0	0	0	0	0	0	0	0	0	11:Phi

To test your understanding, let's consider a couple of more examples. First, consider a model with 2 age-classes for apparent survival, with time-dependence for the first (juvenile) age-class only (adult survival is held constant over time). The PIM structure for φ corresponding to this model is shown at the top of the next page.

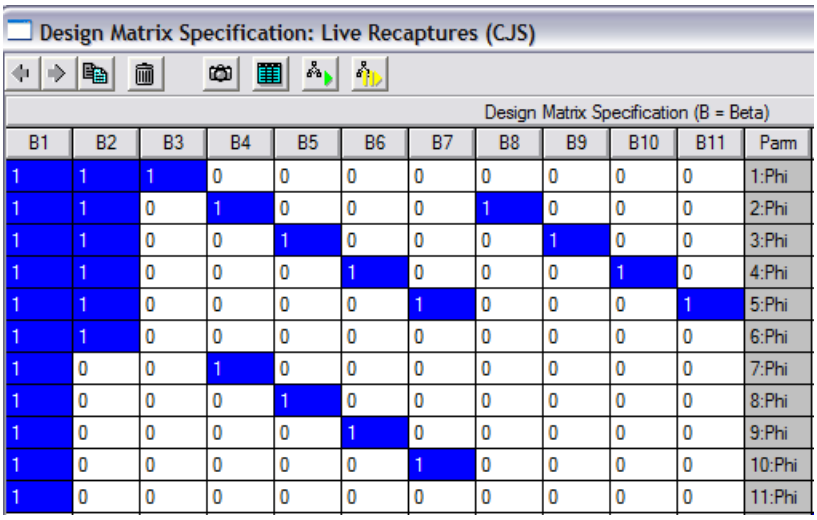
1	7	7	7	7	7
	2	7	7	7	7
		3	7	7	7
			4	7	7
				5	7
					6

Let's first build this model $\{\varphi_{a2-t/.}p_t\}$ using PIMs. By now you should be able to do this on your own, so we'll simply build the model and add the results to the browser:

Results Browser: Live Recaptures (CJS)

Model	AICc	Delta AICc	AICc Weight	Model Likelihood	No. Par.	Deviance	-2Log(L)
{phi(a2 -t/.)p(t) -- PIM}	5053.8746	0.0000	0.51274	1.0000	13	133.0730	5027.7542
{phi(a2 t/t)p(t)} -- PIM}	5055.3628	1.4882	0.24363	0.4752	16	128.5015	5023.1827
{phi(a2 t/t)p(t)} -- DM}	5055.3628	1.4882	0.24363	0.4752	16	128.5015	5023.1827

Now, let's try building this same model, but using a DM approach. Start by retrieving model $\{\varphi_{a2\ t/t}p_t\}$ from the browser – the DM for that model is shown at the top of the next page:



Design Matrix Specification: Live Recaptures (CJS)

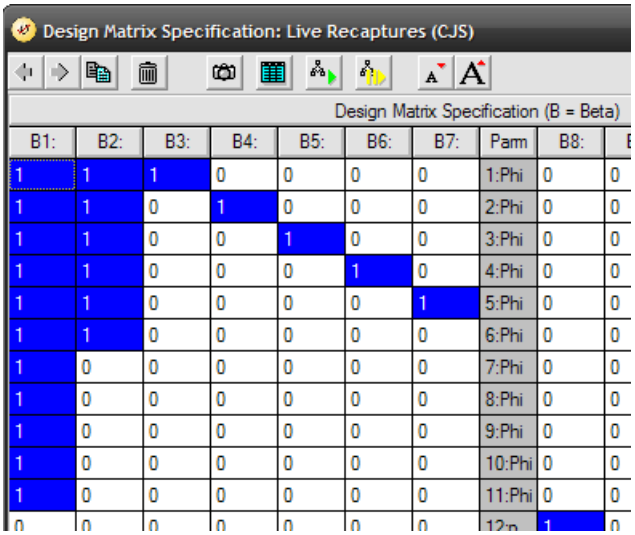
Design Matrix Specification (B = Beta)

B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	B11	Parm
1	1	1	0	0	0	0	0	0	0	0	1:Phi
1	1	0	1	0	0	0	1	0	0	0	2:Phi
1	1	0	0	1	0	0	0	1	0	0	3:Phi
1	1	0	0	0	1	0	0	0	1	0	4:Phi
1	1	0	0	0	0	1	0	0	0	1	5:Phi
1	1	0	0	0	0	0	0	0	0	0	6:Phi
1	0	0	1	0	0	0	0	0	0	0	7:Phi
1	0	0	0	1	0	0	0	0	0	0	8:Phi
1	0	0	0	0	1	0	0	0	0	0	9:Phi
1	0	0	0	0	0	1	0	0	0	0	10:Phi
1	0	0	0	0	0	0	0	0	0	0	11:Phi

For the encounter parameter p, simply use an identity structure (i.e., column of 1's along the diagonal in the part of the DM relating to p).

In this design matrix we have time-variation for both juvenile and adult age classes. However, now we are trying to build model $\{\varphi_{a2t/.}p_t\}$ – time-variation in juvenile survival only. So, we need to modify the DM such that survival is held constant for the adult age class. It might help to consider the linear model for the `TIME` factor, for the two age classes separately. For juveniles, with time variation, the linear model would consist of 5 β terms, coding for the 6 time intervals. In contrast, for adults, survival is constant over time. Recall from chapter 6 that such a constant ('dot') model can be modeled by using a simple 'intercept' – i.e., a single column of 1's.

Keeping this in mind, here is the DM modified to represent model $\{\varphi_{a2-t/.}p_t\}$:



Design Matrix Specification: Live Recaptures (CJS)

Design Matrix Specification (B = Beta)

B1:	B2:	B3:	B4:	B5:	B6:	B7:	Parm	B8:	
1	1	1	0	0	0	0	1:Phi	0	0
1	1	0	1	0	0	0	2:Phi	0	0
1	1	0	0	1	0	0	3:Phi	0	0
1	1	0	0	0	1	0	4:Phi	0	0
1	1	0	0	0	0	1	5:Phi	0	0
1	1	0	0	0	0	0	6:Phi	0	0
1	0	0	0	0	0	0	7:Phi	0	0
1	0	0	0	0	0	0	8:Phi	0	0
1	0	0	0	0	0	0	9:Phi	0	0
1	0	0	0	0	0	0	10:Phi	0	0
1	0	0	0	0	0	0	11:Phi	0	0
0	0	0	0	0	0	0	12:p	1	0

Note that we no longer have any interaction of age with time (since there is no longer time variation for the adults), and that we have eliminated all the time indexing for the adults (parameters $7 \rightarrow 11$). If we run this model, we see that it yields results which are identical to that from the model built using

PIMs (which indicates that the two models are equivalent; i.e., that our DM is correct):

Results Browser: Live Recaptures (CJS)

Model	AICc	Delta AICc	AICc Weight	Model Likelihood	No. Par.	Deviance	-2Log(L)
{phi(a2 t/.)p(t) -- PIM}	5053.8746	0.0000	0.33895	1.0000	13	133.0730	5027.7542
{phi(a2 t/.)p(t)} -- DM}	5053.8746	0.0000	0.33895	1.0000	13	133.0730	5027.7542
{phi(a2 t/t)p(t)} -- PIM}	5055.3628	1.4882	0.16105	0.4751	16	128.5015	5023.1827
{phi(a2 t/t)p(t)} -- DM}	5055.3628	1.4882	0.16105	0.4751	16	128.5015	5023.1827

A final example. Consider a model with 2 age-classes, with time-dependence in each age class. In this example, we have some reason to believe that there is a linear change in survival over time in both age classes (i.e., that survival is varying systematically - increasing or decreasing - over time). How would we fit this model? Well, if you think about it, it is very similar in structure to a classical ANCOVA, except that here our two groups represent the two age classes. In other words, we want to compare the slopes of the linear trend in survival between young and old individuals. If the slopes are not significantly different, we could test against a model with a common slope for both age classes.

First, we need to know the index values of the parameters we're going to constrain. We are going to constrain survival. From the preceding example, we know that for 'juveniles', the parameters of interest are 1 → 6, and for 'adults', the parameters are 7 → 11. So, all we need to do is modify the design matrix we just created accordingly.

How do we handle the vector of increasing values to handle the trend? Recall from Chapter 6 that we could 'model' trend simply by specifying an ordinal sequence of numbers in the design matrix. But, should we use 1 → 6 for juveniles, and 2 → 6 for adults (which corresponds to where the overlap occurs), or is it equivalent to use 1 → 6 and 1 → 5 respectively? As it turns out, it doesn't matter at all to **MARK** - at least not mechanically (i.e., **MARK** will run just fine in either case). However, note that in the second case you would, in effect, be coding each time step differently for each group, which makes the intercepts no longer comparable. Thus, if you choose to accommodate the overlap, the first coding scheme (1 → 6 for juveniles, and 2 → 6 for adults in this example) is preferred. Another possibility, of course, is to drop the first juvenile parameter altogether from our constraint. In other words, to include only those parameters which 'overlap' (i.e., 2 → 6 and 7 → 11).

For our trend analysis, we'll use the former approach, which includes survival for the first age class over the first interval. In this case, the design matrix for the survival elements would look like:

Design Matrix Specification: Li

B1	B2	B3	B4	Parm
1	1	1	1	1:Phi
1	1	2	2	2:Phi
1	1	3	3	3:Phi
1	1	4	4	4:Phi
1	1	5	5	5:Phi
1	1	6	6	6:Phi
1	0	2	0	7:Phi
1	0	3	0	8:Phi
1	0	4	0	9:Phi
1	0	5	0	10:Phi
1	0	6	0	11:Phi

All you need to do now is run **MARK**, and apply the constraint to the underlying model (2 age-classes with time-dependence for survival, simple time-dependence for recaptures). And, as discussed in detail in Chapter 6, by varying which columns of the design matrix you use in the constraint, you can test hypotheses concerning equivalence of slope between age classes, equivalence of intercepts, and so forth. You could also test for additivity (parallelism) in survival for a time-dependent model. Everything covered in Chapter 6 in terms of constraining an underlying CJS model applies equally well to age models (or cohort models, or **any** models).

An important 'nuance' - continuous covariates

In the preceding, we considered the construction of the interaction terms for 2 types of models: one where 'age' and 'time' were both (classification) factors, and one where 'age' interacted with 'time' constrained to follow a linear trend over time. In the first case, we considered the complication that there is no logical interaction of 'age' and 'time' for the first interval, since only the first juvenile age class is present in our marked sample over the first interval (i.e., no adults). In this case, we only included time intervals 2 → 6 in the interaction of 'age' and 'time', as shown in the DM below:

Design Matrix Specification: Live Recaptures (CJS)

Design Matrix Specification (B = Beta)

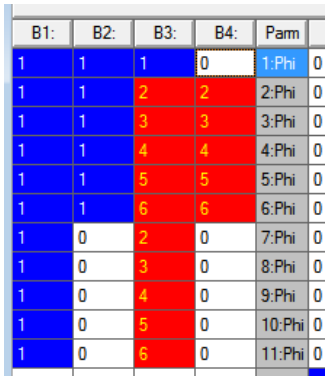
B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	B11	Parm
1	1	1	0	0	0	0	0	0	0	0	1:Phi
1	1	0	1	0	0	0	1	0	0	0	2:Phi
1	1	0	0	1	0	0	0	1	0	0	3:Phi
1	1	0	0	0	1	0	0	0	1	0	4:Phi
1	1	0	0	0	0	1	0	0	0	1	5:Phi
1	1	0	0	0	0	0	0	0	0	0	6:Phi
1	0	0	1	0	0	0	0	0	0	0	7:Phi
1	0	0	0	1	0	0	0	0	0	0	8:Phi
1	0	0	0	0	1	0	0	0	0	0	9:Phi
1	0	0	0	0	0	1	0	0	0	0	10:Phi
1	0	0	0	0	0	0	0	0	0	0	11:Phi

In the second case, where 'time' was constrained to follow a linear trend within each age class, we used the following DM:

Design Matrix Specification: Li

B1	B2	B3	B4	Parm
1	1	1	1	1:Phi
1	1	2	2	2:Phi
1	1	3	3	3:Phi
1	1	4	4	4:Phi
1	1	5	5	5:Phi
1	1	6	6	6:Phi
1	0	2	0	7:Phi
1	0	3	0	8:Phi
1	0	4	0	9:Phi
1	0	5	0	10:Phi
1	0	6	0	11:Phi

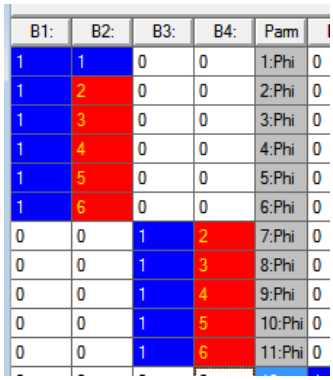
While this seems reasonable enough at first glance, you might now wonder why we included the interaction for the first time interval, since there are only juveniles present during that interval. For example, why didn't we use something like the following:



B1:	B2:	B3:	B4:	Parm	
1	1	1	0	1:Phi	0
1	1	2	2	2:Phi	0
1	1	3	3	3:Phi	0
1	1	4	4	4:Phi	0
1	1	5	5	5:Phi	0
1	1	6	6	6:Phi	0
1	0	2	0	7:Phi	0
1	0	3	0	8:Phi	0
1	0	4	0	9:Phi	0
1	0	5	0	10:Phi	0
1	0	6	0	11:Phi	0

where we put a '0' in the first row of the interaction column? Well, as it turns out, if we try this approach, **MARK** will give us the wrong answer. How do we know this?

Well, let's first start by considering a different approach to modeling trend within each age classes, with interaction between the age classes. Rather than use a common intercept (which we prefer in general because it allows us to build the additive models introduced in Chapter 6), we'll use a DM where each age class has it's own intercept:



B1:	B2:	B3:	B4:	Parm	
1	1	0	0	1:Phi	0
1	2	0	0	2:Phi	0
1	3	0	0	3:Phi	0
1	4	0	0	4:Phi	0
1	5	0	0	5:Phi	0
1	6	0	0	6:Phi	0
0	0	1	2	7:Phi	0
0	0	1	3	8:Phi	0
0	0	1	4	9:Phi	0
0	0	1	5	10:Phi	0
0	0	1	6	11:Phi	0

If we run this model, we see that in fact it yields the same deviance and parameter estimates as the model we fit originally, using a common intercept (i.e., the DM at the bottom of the preceding page).

What is important here is that in this case, '`time`' is no longer a simple, unconstrained factor, but is being constrained as a linear function – in this case, constrained to follow a linear trend. When '`time`' is an unconstrained factor, we need to only include interactions where both '`time`' and '`age`' occur. However, when '`time`' is constrained to be a linear function of something (say, a trend, or some environmental covariate), then we need to include all time steps in the interaction.

We can demonstrate this by means of another example. Suppose that instead of constraining variation in 'time' to follow a trend, we constrain variation to be a linear function of some environmental covariate (say, average temperature). Suppose that the average temperature over each of 6 intervals is: $\{10, 14, 22, 11, 15, 17\}$.

Here is the DM using a separate intercept for each age class, where survival within a given age class is constrained to be a function of the average temperature:

B1:	B2:	B3:	B4:	Parm	
1	10	0	0	1:Phi	0
1	14	0	0	2:Phi	0
1	22	0	0	3:Phi	0
1	11	0	0	4:Phi	0
1	15	0	0	5:Phi	0
1	17	0	0	6:Phi	0
0	0	1	14	7:Phi	0
0	0	1	22	8:Phi	0
0	0	1	11	9:Phi	0
0	0	1	15	10:Phi	0
0	0	1	17	11:Phi	0

Model deviance for this DM fit to the data (with full time-dependence in p) is 5102.437. Now, if we try a DM with separate intercepts for each age class, and leave out the interaction for the first time interval for juveniles

B1:	B2:	B3:	B4:	Parm
1	1	10	0	1:Phi
1	1	14	14	2:Phi
1	1	22	22	3:Phi
1	1	11	11	4:Phi
1	1	15	15	5:Phi
1	1	17	17	6:Phi
1	0	14	0	7:Phi
1	0	22	0	8:Phi
1	0	11	0	9:Phi
1	0	15	0	10:Phi
1	0	17	0	11:Phi

yields a model deviance of 5088.601, which is not the same. And, clearly, then, it isn't the same model we constructed initially using separate intercepts for each age class.

However, if we include the first interval for juveniles in the interaction

B1:	B2:	B3:	B4:	Parm	
1	1	10	10	1:Phi	0
1	1	14	14	2:Phi	0
1	1	22	22	3:Phi	0
1	1	11	11	4:Phi	0
1	1	15	15	5:Phi	0
1	1	17	17	6:Phi	0
1	0	14	0	7:Phi	0
1	0	22	0	8:Phi	0
1	0	11	0	9:Phi	0
1	0	15	0	10:Phi	0
1	0	17	0	11:Phi	0

the model deviance is 5102.437, identical to the model we fitting using a DM with separate intercepts.

The reason for the difference here – needing to include the covariate interaction for all time intervals is we need to fully specify the linear relationship between the parameter and the covariate. Leaving out one covariate for a particular age class, and replacing it with a 0, would change the nature of the linear model for the covariate. In contrast, for 'time' as a factor, there is no underlying linear model for time. This is a subtle point of distinction, and one you need to think about a bit, whenever you have a 'grouping' factor (like 'age') with unequal number of time intervals (say, between juveniles and adults).

7.2.1. DM with > 2 age classes: 'ugly' interaction terms

As noted in the preceding examples involving 2 age classes, the key consideration to building the design matrix for age models in general are (i) keeping track of the asymmetry of which age classes in the marked sample are represented at each occasion, and (ii) how to handle the interactions among age classes. These challenges are compounded with > 2 age classes.

Consider the following PIM:

1	7	12	13	14	15
	2	8	13	14	15
		3	9	14	15
			4	10	15
				5	11
					6

Here, we see we have specified 3 age classes with time-dependence for (say) φ: parameters $1 \rightarrow 6$ for the first age class (for the first 6 time intervals), parameters $7 \rightarrow 11$ for the second age class (for time intervals $2 \rightarrow 6$), and parameters $12 \rightarrow 15$ for the third and final age class (for time intervals $3 \rightarrow 6$). So, first thing to consider, with > 2 age classes, we need to have > 1 term in the linear model to code for age: $(k-1) = 3-1 = 2$ terms, in fact. For 7 occasions (6 time intervals) we need $(k-1) = (6-1) = 5$ terms for time. Thus, our linear model starts with the following structure

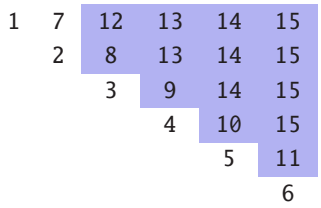
$$\text{logit}(\hat{\varphi}) = \beta_1 + \beta_2(\texttt{AGE}_1) + \beta_3(\texttt{AGE}_2) + \beta_4(\texttt{T}_1) + \beta_5(\texttt{T}_2) + \beta_6(\texttt{T}_3) + \beta_7(\texttt{T}_4) + \beta_8(\texttt{T}_5) + \text{???}$$

What about the interactions? Here, we need to be careful. We will start by having a look at the PIM – what are the intervals for which there are interactions between age 1 and age 2 individuals?

1	7	12	13	14	15
	2	8	13	14	15
		3	9	14	15
			4	10	15
				5	11
					6

The shaded parts of the PIM indicate clearly where these interactions occur - in intervals $(2 \rightarrow 6)$ - there is no interaction of age 1 and age 2 individuals in the marked population for interval 1 since there are no age 2 individuals in interval 1! This is precisely the same situation we saw earlier for a simpler 2 age-class model.

What about for the interactions between 2 and 3+ year old individuals? Again, have a careful look at the PIM:



The shaded parts of the PIM indicate where (i.e., over which time intervals) the interactions of 2 and 3+ year old individuals occur - in intervals $(3 \rightarrow 6)$.

Next, we add terms to the linear model to reflect *plausible* interactions between the age classes. Start by considering the interaction among 'age' and 'time' for the interaction of 1 and 2 year-old individuals. We enter a term in the linear model for each of the intervals for which there is a logical (possible) interaction of these two age classes (intervals $(2 \rightarrow 6)$; $k = 5$ intervals, 4 β terms):

$$\begin{aligned}\text{logit}(\hat{\varphi}) = {} & \beta_1 + \beta_2(\texttt{AGE}_1) + \beta_3(\texttt{AGE}_2) + \beta_4(\texttt{T}_1) + \beta_5(\texttt{T}_2) + \beta_6(\texttt{T}_3) + \beta_7(\texttt{T}_4) + \beta_8(\texttt{T}_5) \\ & + \beta_9(\texttt{AGE}_1 \cdot \texttt{T}_2) + \beta_{10}(\texttt{AGE}_1 \cdot \texttt{T}_3) + \beta_{11}(\texttt{AGE}_1 \cdot \texttt{T}_4) + \beta_{12}(\texttt{AGE}_1 \cdot \texttt{T}_5) \\ & + \ ???\end{aligned}$$

Now, what about the interactions of age 2 and 3+ individuals? As before, we enter a term in the linear model for each of the intervals for which there is a logical (possible) interaction of these two age classes (in this case, intervals $(3 \rightarrow 6)$; $k = 4$ intervals, $k - 1 = (4\text{-}1) = 3$ β terms):

$$\begin{aligned}\text{logit}(\hat{\varphi}) = {} & \beta_1 + \beta_2(\texttt{AGE}_1) + \beta_3(\texttt{AGE}_2) + \beta_4(\texttt{T}_1) + \beta_5(\texttt{T}_2) + \beta_6(\texttt{T}_3) + \beta_7(\texttt{T}_4) + \beta_8(\texttt{T}_5) \\ & + \beta_9(\texttt{AGE}_1 \cdot \texttt{T}_2) + \beta_{10}(\texttt{AGE}_1 \cdot \texttt{T}_3) + \beta_{11}(\texttt{AGE}_1 \cdot \texttt{T}_4) + \beta_{12}(\texttt{AGE}_1 \cdot \texttt{T}_5) \\ & + \beta_{13}(\texttt{AGE}_2 \cdot \texttt{T}_3) + \beta_{14}(\texttt{AGE}_2 \cdot \texttt{T}_4) + \beta_{15}(\texttt{AGE}_2 \cdot \texttt{T}_5)\end{aligned}$$

Now, if you've been paying attention, you might have noticed that the subscripting for AGE for these new β terms changed from '1' (for the age 1 and age 2 interaction terms) to '2' (for the age 2 and age 3+ interaction terms). Why? Well, you might guess with a bit of thought. Remember that we have 3 age classes here, so we need 2 columns of dummy variables to code for it. If we use

$$\beta_2 = \begin{cases} \mathbf{1} \text{ if age} = 1 \\ \mathbf{0} \text{ if other} \end{cases} \qquad \beta_3 = \begin{cases} \mathbf{1} \text{ if age} = 2 \\ \mathbf{0} \text{ if other} \end{cases}$$

then $\texttt{AGE}_{3+}$ is the reference age class (for individuals age $\geq$ 3 years), $\texttt{AGE}_1$ corresponds to age 1 year individuals, and $\texttt{AGE}_2$ corresponds to age 2 year individuals. So, for the first interaction, you're multiplying the column for β_2 against the (5-1)=4 columns for those time intervals where the interaction between age 1 and 2 occurs, which yields 4 β terms for that interaction. For the second interaction, you're multiplying the column for β_3 against the (4-1)=3 columns for those time intervals where the interaction between 2 and 3+ occurs, which yields 3 additional β terms for that interaction.

The dummy variable coding for AGE also indicates why there is no interaction of 1 year old and 3+ year old individuals, and no 3-way interaction of (age 1 × age 2 × age 3+) individuals - even though these interactions are plausible for some intervals. Why? Simple – the product of 0 with anything is 0,

so these terms disappear from the model (remember, age 3+ individuals are coded as '0' for β_2 and '0' for β_3). Here is the design matrix (where column B1 corresponds to β_0 in the linear model):

Design Matrix Specification: Live Recaptures (CJS)

Design Matrix Specification (B = Beta)

B1 int	B2 age1	B3 age2	B4 t1	B5 t2	B6 t3	B7 t4	B8 t5	B9 a1t2	B10 a1t3	B11 a1t4	B12 a1t5	B13 a2t3	B14 a2t4	B15 a2t5	Parm
1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	1:Phi
1	1	0	0	1	0	0	0	1	0	0	0	0	0	0	2:Phi
1	1	0	0	0	1	0	0	0	1	0	0	0	0	0	3:Phi
1	1	0	0	0	0	1	0	0	0	1	0	0	0	0	4:Phi
1	1	0	0	0	0	0	1	0	0	0	1	0	0	0	5:Phi
1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	6:Phi
1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	7:Phi
1	0	1	0	0	1	0	0	0	0	0	0	1	0	0	8:Phi
1	0	1	0	0	0	1	0	0	0	0	0	0	1	0	9:Phi
1	0	1	0	0	0	0	1	0	0	0	0	0	0	1	10:Phi
1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	11:Phi
1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	12:Phi
1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	13:Phi
1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	14:Phi
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	15:Phi

Make sure you see the correspondence between this design matrix and the linear model we're fitting:

$$\begin{aligned}\text{logit}(\hat{\varphi}) = {} & \beta_1 + \beta_2(\text{AGE}_1) + \beta_3(\text{AGE}_2) + \beta_4(\text{T}_1) + \beta_5(\text{T}_2) + \beta_6(\text{T}_3) + \beta_7(\text{T}_4) + \beta_8(\text{T}_5) \\ & + \beta_9(\text{AGE}_1 \cdot \text{T}_2) + \beta_{10}(\text{AGE}_1 \cdot \text{T}_3) + \beta_{11}(\text{AGE}_1 \cdot \text{T}_4) + \beta_{12}(\text{AGE}_1 \cdot \text{T}_5) \\ & + \beta_{13}(\text{AGE}_2 \cdot \text{T}_3) + \beta_{14}(\text{AGE}_2 \cdot \text{T}_4) + \beta_{15}(\text{AGE}_2 \cdot \text{T}_5)\end{aligned}$$

Now, for a real test – how would the design matrix change if you had made age 1 the reference age class in your dummy variable coding scheme?

$$\beta_2 = \begin{cases} \mathbf{1} \text{ if age} = 2 \\ \mathbf{0} \text{ if other} \end{cases} \qquad \beta_3 = \begin{cases} \mathbf{1} \text{ if age} = 3 \\ \mathbf{0} \text{ if other} \end{cases}$$

Would the linear model itself change, or just the structure of the design matrix (or both)? We'll leave that to you as an 'exercise'. Of course, you should realize by now that while changing the reference age class will change your interpretation of the β's, it won't change the overall fit of the model to the data.

7.3. Using data where both young and adults are marked

Everything we have covered concerning age models has to this point assumed that we were dealing with individuals all marked as young (or, at the least, at a common age). However, very commonly when we take samples from populations we sample individuals from several age classes, and individually mark any unmarked individuals. For example, if our sample contains both newborns and adults, we will mark both age classes. Of course, we could choose to just mark the young, but this is often not desirable. Analytically, the question becomes - how to deal with data from both groups of newly marked

individuals? In fact, the use of the word 'group' was intentional - the solution is simply to treat both types of individuals as different groups with **MARK**.

Previously, we have compared males and females, or controls and treatments, or good and poor colonies. In this case we simply have 'marked as young' versus 'not marked as young'. In this case, we are testing the hypothesis that survival within an age-class over a given interval doesn't depend upon the age at which the individual was marked.

How do we do this in **MARK**? Easy - in fact, it requires only a slightly more involved application of what we've already done. In effect, the analysis becomes one of comparing survival over a given interval between these two 'groups' (marked as young and marked as adult)- we condition our data based on the age at which the individual was marked: young or adult. For the birds marked as young, we clearly anticipate the possibility of age-specific differences in survival. Let's assume time-dependence in survival in the first age-class (spanning one year), with time variation in the adult age classes - for this group (i.e., marked as young). Assume 5 occasions. Thus, our PIM for birds marked as young would have the following structure:

1	5	6	7
	2	6	7
		3	7
			4

But what about the individuals marked *as* adults? Well, for adults we assume simple time-dependence. We don't expect any age-specificity in adult survival. So, our PIM file for adults would be:

8	9	10	11
	9	10	11
		10	11
			11

What comes next? Well, basically what we want to determine is whether or not the 'adult survival' differs between the 2 groups. In other words, do the estimates for parameters $5 \rightarrow 7$ (adult survival probabilities for birds marked as young) differ significantly from estimates for parameters $9 \rightarrow 11$ (adult survival for individuals marked as adults). Why is parameter 8 not included in our comparison? If you think about it for a moment, you'll realize why. Parameter 8 is the probability of a bird marked as an adult at occasion 1 surviving to occasion 2. It cannot be compared with the estimate for parameter 1, which is the survival of an individual marked as a juvenile at occasion 1 to occasion 2. So, we would be comparing 'apples and oranges' - a juvenile survival probability with an adult survival probability. By now, you should know exactly how to do this comparison. In fact, you should realize that there are several ways you could do this - you could modify the PIMs, or modify the design matrix. The most straightforward approach is to modify the PIMs - simply setting the various parameters equal to each other, by using the same index value. Which parameters? $5 = 9, 6 = 10, 7 = 11$. So, the PIMs would then look like:

1	5	6	7
	2	6	7
		3	7
			4

8	5	6	7
	5	6	7
		6	7
			7

Got it? Make sure you do! Notice that there the main difference is that with individuals marked as both young, and adults, there are both young and adults in the first interval. Thus, the parameter indexing in the PIMs must (in general) reflect this - note that we've used 1 for marked as young, and 8 for marked as adults. The actual indexing (numbering) isn't as important as the structural differences in the PIMs. Since it is very common to work with data containing individuals marked both as young, and as adults, it's important you truly understand what we're doing here.

7.3.1. Marked as young and adult: the design matrix. . .

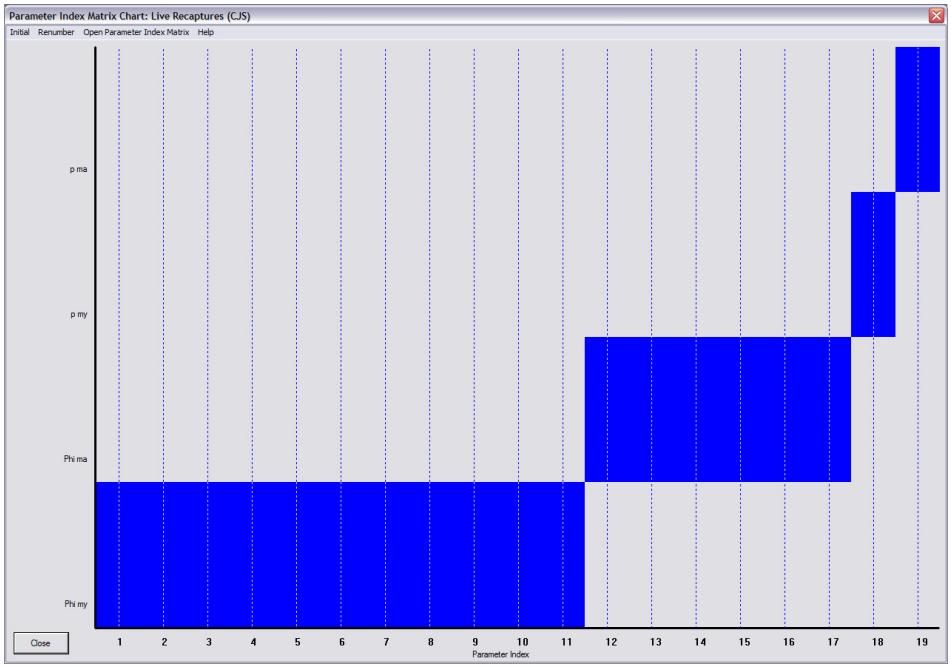
Now, for the **big** test of your understanding of design matrices, and age models. Consider the following problem - again, we focus on a situation where individuals are marked as both young and adults. Assume that our general model is a model with 2 age classes for individuals marked as young, 1 age class for individuals marked as adults. Assume full time-dependence for apparent survival, with no time variation in encounter probability between adults or young, but different encounter probabilities between the two groups. Assume we have 7 occasions. The data for this 'test' are contained in `AGE_YA.INP`.

Now, we could build our general model using PIMs - here are the apparent survival PIMs, for marked as juveniles, and marked as adults, respectively.

1	7	8	9	10	11
	2	8	9	10	11
		3	9	10	11
			4	10	11
				5	11
					6

12	13	14	15	16	17
	13	14	15	16	17
		14	15	16	17
			15	16	17
				16	17
					17

Here is the corresponding PIM chart:



In the PIM chart, parameters $1 \rightarrow 6$ represent first-year survival among those marked as young, parameters $7 \rightarrow 11$ corresponds to adult survival for individuals marked as young (remember, for such individuals, there is no adult survival for the first interval, since there are no adults yet!). Parameters $12 \rightarrow 17$ correspond to adult survival for individuals marked as adults, and (finally) parameters 18 and 19 correspond to encounter probability for individuals marked as young, and adult, respectively. Got it? Make sure you do.

Now, while this is all well and good, and we could build our general model this way, using the PIM chart, we realize by now (you should!) that this is ultimately limiting, since there are some models we can't build using the PIM's. For example, additive models (such as, a model where adult survival and offspring survival for individuals marked as young differed by an additive constant). So, in general, it is recommended you try to build your general model using the design matrix. Your challenge, then, is how to build a design matrix which corresponds to this PIM chart.

Actually, it's not too bad, if you think it through logically. First, we'll start with writing out the linear model for survival: it's pretty ugly (since it has 17 terms), but it is a useful exercise. We'll let `M` represent 'age at marking' (one level of grouping), and `A` represent the age of the individual (note that `A` is nested within `M`). As per usual, we'll let `T` represent `TIME`.

To help you follow along, we've put main effects and each of the two interactions on separate lines of the equation.

$$\begin{aligned}\text{logit}(\hat{\varphi}) = {} & \beta_1 + \beta_2(\texttt{M}) + \beta_3(\texttt{A}) + \beta_4(\texttt{T}_1) + \beta_5(\texttt{T}_2) + \beta_6(\texttt{T}_3) + \beta_7(\texttt{T}_4) + \beta_8(\texttt{T}_5) \\ & + \beta_9(\texttt{M.T}_2) + \beta_{10}(\texttt{M.T}_3) + \beta_{11}(\texttt{M.T}_4) + \beta_{12}(\texttt{M.T}_5) \\ & + \beta_{13}(\texttt{A.T}_1) + \beta_{14}(\texttt{A.T}_2) + \beta_{15}(\texttt{A.T}_3) + \beta_{16}(\texttt{A.T}_4) + \beta_{17}(\texttt{A.T}_5)\end{aligned}$$

A couple of comments are needed here. First, why no '3-way interaction term' (i.e., why no `M.A.T` columns?). Simple - there are no young individuals within those marked as adults, so there is no logical 3-way interaction. The same logic applies for 'why no (`A.M`) column?'.

Second, why do the interactions of age at marking (`M`) and time (`T`) start with the second time interval, whereas the interaction of age within age at marking (`A`) and time (`T`) start with the first time interval? This is a subtle point - but an important one. If you look closely at the PIM structure for marked as juveniles, and as adults, respectively

1	7	8	9	10	11
	2	8	9	10	11
		3	9	10	11
			4	10	11
				5	11
					6

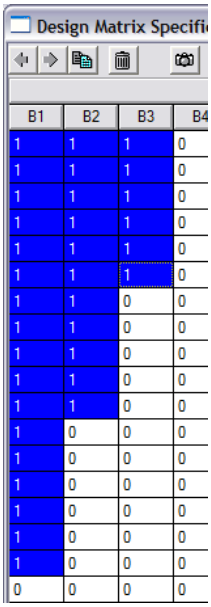
12	13	14	15	16	17
	13	14	15	16	17
		14	15	16	17
			15	16	17
				16	17
					17

you'll see that there are logical interactions of age (juvenile or adult) for all time intervals (thus, interactions for $\texttt{A.T}_1 \rightarrow \texttt{A.T}_5$), while for age at marking, there are really interactions starting only with the second interval - there is no logical interaction for parameters 1 and 12 in the PIMs (thus, interactions for $\texttt{M.T}_2 \rightarrow \texttt{M.T}_5$). This is a bit tricky, so make sure you understand it.

OK, given this linear model, let's start building our design matrix. We know we need an intercept column (representing β_0). Then, a column indicating whether the individuals were marked as young, or adults (representing β_1). Then (and perhaps the only tricky bit) another column which indicates the 'age class' (young or adult) within 'marking' group (for example, young or adult among those marked

as young, and obviously adult only for those marked as adults). This represents β_2 in the linear model.

Here is our DM so far – make sure you study it carefully.



Here, column `B1` is the intercept, `B2` is a dummy variable for 'marking' group: marked as young (first eleven `1`'s) or marked as adult (the next six `0`'s), and `B3` is a column indicating 'age class' (young or adult) within 'marking' - we've used `1` for young, and `0` for adults. Obviously, marked as adult individuals are going to be all `0`'s.

What comes next? Well, our general model has time-dependence for all groups/age classes, so we need to add some dummy variables for time. Look closely again at the linear model:

$$\begin{aligned}\text{logit}(\hat{\varphi}) = \beta_1 + \beta_2(\mathtt{M}) + \beta_3(\mathtt{A}) + \beta_4(\mathtt{T}_1) + \beta_5(\mathtt{T}_2) + \beta_6(\mathtt{T}_3) + \beta_7(\mathtt{T}_4) + \beta_8(\mathtt{T}_5) \\ + \beta_9(\mathtt{M.T}_2) + \beta_{10}(\mathtt{M.T}_3) + \beta_{11}(\mathtt{M.T}_4) + \beta_{12}(\mathtt{M.T}_5) \\ + \beta_{13}(\mathtt{A.T}_1) + \beta_{14}(\mathtt{A.T}_2) + \beta_{15}(\mathtt{A.T}_3) + \beta_{16}(\mathtt{A.T}_4) + \beta_{17}(\mathtt{A.T}_5)\end{aligned}$$

In the linear model, we have five β terms to code for the 6 intervals (remember, $n-1$ columns). Now, we've seen time coding for 'age' models several times already, so you might think this should be fairly straightforward. It is - but you need to be careful, and think about when the time intervals start for individuals of a given age, as a function of the age at which they were marked.

If you think about it for a moment (or two), you should realize that the time indexing for the adult age class for individuals marked as young starts in the second interval, not the first. Again, this is because for such individuals (marked as young) there are no adults in the first interval (obviously - if they're marked as young, they can't be adults). But, for adults marked as adults, all 6 intervals in the 7 occasion study are represented. Thus, the time coding is the same for first-year survival of individuals marked as young, and adult survival for individuals marked as adults. The design matrix with the time-coding entered is shown at the top of the next page. Make sure you understanding the coding for each level of `A` and `M`.

Design Matrix Specification: Live Recaptures (CJS)

B1	B2	B3	B4	B5	B6	B7	B8	Parm
1	1	1	1	0	0	0	0	1:Phi
1	1	1	0	1	0	0	0	2:Phi
1	1	1	0	0	1	0	0	3:Phi
1	1	1	0	0	0	1	0	4:Phi
1	1	1	0	0	0	0	1	5:Phi
1	1	1	0	0	0	0	0	6:Phi
1	1	0	0	1	0	0	0	7:Phi
1	1	0	0	0	1	0	0	8:Phi
1	1	0	0	0	0	1	0	9:Phi
1	1	0	0	0	0	0	1	10:Phi
1	1	0	0	0	0	0	0	11:Phi
1	0	0	1	0	0	0	0	12:Phi
1	0	0	0	1	0	0	0	13:Phi
1	0	0	0	0	1	0	0	14:Phi
1	0	0	0	0	0	1	0	15:Phi
1	0	0	0	0	0	0	1	16:Phi
1	0	0	0	0	0	0	0	17:Phi

Now, for the final steps – building the interaction terms. The key to remember is that (i) we have two grouping variables (age of marking, and age-class within age of marking), and (ii) we can't have an interaction between a grouping variable and a time-period, if the time-period is not logically part of the group. In this case, it makes no sense to have an interaction of time and age for interval 1 for individuals marked as young, since there are no adults in interval 1 for individuals marked as young. Thus, for these individuals, the interaction terms start with the first interval where both age classes are represented. Got it? Well, it might help to see the full design matrix, with the interaction terms in place. Here it is:

Design Matrix Specification: Live Recaptures (CJS)

Design Matrix Specification (B = Beta)

B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	B11	B12	B13	B14	B15	B16	B17	Parm	B18	B19
1	1	1	1	0	0	0	0	0	0	0	0	1	0	0	0	0	1:Phi	0	0
1	1	1	0	1	0	0	0	1	0	0	0	0	1	0	0	0	2:Phi	0	0
1	1	1	0	0	1	0	0	0	1	0	0	0	0	1	0	0	3:Phi	0	0
1	1	1	0	0	0	1	0	0	0	1	0	0	0	0	1	0	4:Phi	0	0
1	1	1	0	0	0	0	1	0	0	0	1	0	0	0	0	1	5:Phi	0	0
1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	6:Phi	0	0
1	1	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	7:Phi	0	0
1	1	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	8:Phi	0	0
1	1	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	9:Phi	0	0
1	1	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	10:Phi	0	0
1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	11:Phi	0	0
1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	12:Phi	0	0
1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	13:Phi	0	0
1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	14:Phi	0	0
1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	15:Phi	0	0
1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	16:Phi	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	17:Phi	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	18:p	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	19:p	0	1

Again, it is worth referring to the linear model we're trying to build:

$$\begin{aligned}\text{logit}(\hat{\varphi}) = {} & \beta_1 + \beta_2(\texttt{M}) + \beta_3(\texttt{A}) + \beta_4(\texttt{T}_1) + \beta_5(\texttt{T}_2) + \beta_6(\texttt{T}_3) + \beta_7(\texttt{T}_4) + \beta_8(\texttt{T}_5) \\ & + \beta_9(\texttt{M.T}_2) + \beta_{10}(\texttt{M.T}_3) + \beta_{11}(\texttt{M.T}_4) + \beta_{12}(\texttt{M.T}_5) \\ & + \beta_{13}(\texttt{A.T}_1) + \beta_{14}(\texttt{A.T}_2) + \beta_{15}(\texttt{A.T}_3) + \beta_{16}(\texttt{A.T}_4) + \beta_{17}(\texttt{A.T}_5)\end{aligned}$$

The interaction of 'age of marking' (`M`, in column `B2`) and time (`B4` → `B8`) is shown in columns `B9` → `B12`. Note that the coding indicates that the interactions start with interval 2. The reasons for this were presented in the previous section (7.1).

The interaction of 'age within age of marking' (`A`, in column `B3`) and time is shown in columns `B13` → `B17`. The two blue boxes along the diagonal in the lower right corner (columns B18 and B19) code for the group-specific recapture probabilities, for marked as young and marked as adults, respectively - single values since we're assuming only differences between age of marking classes, but no time variation.

Study the preceding design matrix carefully. If you understand it, great! This is perhaps one of the more difficult design matrices you can build - but a very common one. If you don't understand it, go back through the preceding page or so, and try again (and, failing that, work through Chapter 6 again). Understanding how to construct design matrices in general is critical to understanding how to use **MARK** at a high level.

OK, to make sure, a final test (and you thought we were done...). Same data set as above, but now you want to build a particular reduced model, corresponding to the following PIM structure for survival (again we assume 7 occasions in the study):

1	7	8	9	10	11
	2	8	9	10	11
		3	9	10	11
			4	10	11
				5	11
					6

12	7	8	9	10	11
	7	8	9	10	11
		8	9	10	11
			9	10	11
				10	11
					11

The structure of the first PIM (representing individuals marked as young) should be familiar - the diagonal represents the first year after marking, and the off-diagonal represents time-varying survival for these individuals as adults. The second PIM (representing marked as adults) requires a bit of care. Here, we use the same indexing (7 → 11) in both PIMs, for adult survival for the 2nd through 6th intervals. But, what about the use of the index 12 in the first column? Well, if you think about it, this should make some sense – we can't use a 1, since we've used that for first-year survival in the other PIM for individuals marked as young. So, we need to come up with a coding reflecting that for those marked as adults, we do in fact have an adult survival estimate over the first interval. But, since there are no adults in that interval for individuals marked as young, we can't constrain the two PIMs to have the same indexing for that interval. If this isn't clear, go through it again.

Again, we can build this model using the PIMs themselves, but part of our motivation in the first place for building the design matrix corresponding to our most general model is that doing so makes it easier for us to build reduced models (like the one we're considering) - all we need to do is figure out which of the columns in the design matrix for the general model we need to modify (typically, by deleting one or more columns). Look closely again at the design matrix for our general model (at the bottom of the preceding page). Again, the interaction of 'age of marking' (`M`, in column `B2`) and time (`B4` → `B8`) is shown in columns `B9` → `B12`. The interaction of 'age within age of marking' (`A`, in column `B3`)

and time is shown in columns B13 → B17.

So, how do we build our reduced model, where adult survival for intervals 2 to 6 is constrained to be the same for individuals marked either as young or adults? Easy, if you understand the preceding design matrix. We see immediately that our reduced model does **not** have an interaction between time and 'age at marking'. So, we simply (1) delete the column corresponding to the 'age at marking' factor (i.e., column 2, B2), and (ii) delete the columns corresponding to the interaction of 'age at marking' and time. Test this for yourself - build the reduced model using PIMs, then build it using a design matrix approach, by deleting the appropriate interaction columns from the design matrix we constructed for our general model (shown above). You'll see that you get the same results (as you should!).

However, the alert reader might notice that the indexing we've used in this example (shown in the PIMs, above) is somewhat 'awkward', in the sense that parameter 12 corresponds to adult survival for the *first* interval, whereas parameters 7 → 11 correspond to the later intervals 2 → 6, respectively. The reason we have this 'problem' (which is a problem only in the sense that you'd need to remember that parameter 12 corresponds to a time interval before parameters 7 → 11) is because we made the first group 'marked as young', and the second group 'marked as adults' in the `.inp` file. If in fact we'd set up the `.inp` file the other way around (marked as adults as the first frequency column, and marked as young as the second frequency column), then we could use the following, more 'logical' PIM indexing.

1	2	3	4	5	6
	2	3	4	5	6
		3	4	5	6
			4	5	6
				5	6
					6

7	2	3	4	5	6
	8	3	4	5	6
		9	4	5	6
			10	5	6
				11	6
					12

Here, we see that parameters 1 → 6 correspond to adult survival, while parameters 7 → 12 correspond to juvenile survival. Here is the DM corresponding to these modified PIMs:

Design Matrix Specification: Live Recaptures (CJS)

B1:	B2:	B3:	B4:	B5:	B6:	B7:	B8:	B9:	B10:	B11:	B12:	Parm
1	1	1	0	0	0	0	1	0	0	0	0	1:Phi
1	1	0	1	0	0	0	0	1	0	0	0	2:Phi
1	1	0	0	1	0	0	0	0	1	0	0	3:Phi
1	1	0	0	0	1	0	0	0	0	1	0	4:Phi
1	1	0	0	0	0	1	0	0	0	0	1	5:Phi
1	1	0	0	0	0	0	0	0	0	0	0	6:Phi
1	0	1	0	0	0	0	0	0	0	0	0	7:Phi
1	0	0	1	0	0	0	0	0	0	0	0	8:Phi
1	0	0	0	1	0	0	0	0	0	0	0	9:Phi
1	0	0	0	0	1	0	0	0	0	0	0	10:Phi
1	0	0	0	0	0	1	0	0	0	0	0	11:Phi
1	0	0	0	0	0	0	0	0	0	0	0	12:Phi
0	0	0	0	0	0	0	0	0	0	0	0	13:p
0	0	0	0	0	0	0	0	0	0	0	0	14:p

Column 1 is the intercept, column 2 is 'age class' (i.e., adult or juvenile - here we've used the '1' to

indicate adult, 0 to indicate juvenile), followed by columns 3 → 7 for time, and columns 8 → 12 for the (age × time) interactions. Again, the point of this somewhat extended exercise is that the indexing in the PIMs in this example is affected by the order in which the 'groups' (marked as adults, marked as young) are entered in the `.inp` file.

7.4. 'Time since marking' - when an age model is NOT an 'age' model

Recall from the preamble to this chapter that 'age' models are motivated by the fact that we expect that individuals of different ages might have different survival or recapture probabilities. Also recall that we noted that within a cohort, age and time were synonymous (and thus, age and time effects could not be separately estimated, at least within a cohort). The key here is that 'age=time'. In fact, the age of an animal is always defined as the time since the animal was marked. If the animal was marked as a newborn, then the time since marking is equivalent to true chronological age. But, what if the animal was marked as an adult-say, as a 3 year old? If the year of marking is year (t), and it is currently year (t+5), then the animal is chronologically 8 years old, but it is 5 years old, relative to the age at which is was marked.

Why do we care? We care because in many instances, we do not expect to detect true age differences, but rather, differences among individuals as a function of the relative time since marking. Recall that in a true age model, we anticipate that the survival (for example) of newborns might be different (typically lower) than the survival of adults. Structurally, this is equivalent to saying that we anticipate that the survival the year after marking (which in this case refers to the first year of life) will differ from the survival in subsequent years of life (i.e., among those individuals surviving the first year of life, we expect these surviving individuals - now adults - to have different survival in all subsequent years). So, in essence, we are considering variation in survival (in this example) as a function of 'time since marking' (which we will refer to as '*TSM models*'). So, hereafter, we distinguish between 'true' age models (where we are interested in true, chronological age effects) from TSM models, where we're not interested in age, *per se*, but the time elapsed since the animal was marked (i.e., relative age).

A reasonable question at this stage is, 'Why do we need to consider TSM models at all?'. As mentioned, there are a number of common situations where it is the time since marking which influences various parameters. We'll introduce one fairly well-studied example to illustrate the use of TSM models: the presence of *transient* individuals in marked samples. The consideration of transience in marked samples begins with the seminal paper by Pradel and colleagues.* We follow Pradel *et al*. and define a transient individual as 'an individual that is marked, released, and which then permanently emigrates from the sample, such that it is no longer available for encounter in the future'. In other words, a transient is an individual that is seen once (the marking event), then never seen again, because it has emigrated from the population. Recall that in a classic live encounter study, we cannot separate true death from permanent emigration (they are confounded). Thus, a transient individual, which by definition permanently emigrates from the population, will appear to have 'died'. (We realize that some of the more biologically-minded readers will at this point be wondering about 'temporary emigration' - the case where an individual leaves the sample, but not permanently. Temporary emigration is dealt with in a subsequent chapter (15) on the 'robust design'.

We differentiate transient individuals (as defined previously) from resident individuals, which are those individuals that are marked, released, and stay in the sample. Conditional on remaining alive,

* Pradel, R., J. E. Hines, J-D. Lebreton, and J. D. Nichols. 1997. Capture-recapture survival models taking account of transients. *Biometrics* **53**:60-72.

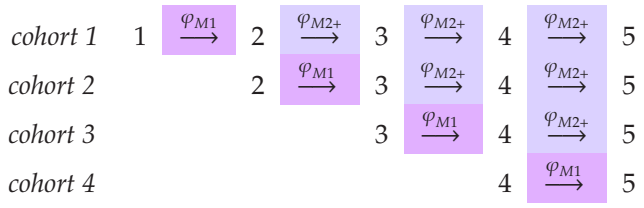
these resident individuals thus have the potential to be encountered again, after the initial marking event. The presence of both resident and transient individuals in the sample is a clear violation of one of the assumptions of the classical CJS model - namely, that all individuals have the same probability of subsequent encounter. The presence of residents and transients violates this assumption, since transients (which have an encounter probability after marking of 0) do not have the same probability of encounter as do residents. Thus, the population is said to show heterogeneity among individuals, in one or more parameters. Earlier in the chapter, we allowed for differences among individuals in terms of true chronological age. But, how do we account for differences among residents and transients in the present example?

The clue to answering this is to recall that a transient is an individual which permanently emigrates after marking. If we start by assuming that this emigration 'event' occurs during the interval after marking, we might start to see a connection with TSM models. If not, the following numerical example should make this clear. Suppose that we have a population of some bird species, where the annual survival of residents is a constant, $\varphi_R = 0.8$. Suppose that the encounter probability for this population is also constant, and equal to 1.0 (i.e., there is perfect, complete registration of all living resident individuals). Now, what about transients? Well, if a transient permanently emigrates immediately after marking, then its apparent survival probability is $\varphi_T = 0$. Now, imagine we start with a sample of 100 individuals from some population. Assume that they are all adults of the same age at the time of marking (thus, no heterogeneity in true age). Suppose that 50% of them are resident, and 50% are transient. Of course, in the 'real world' we wouldn't know this, but for the moment, we'll assume that we do. So, $\mathbf{R}_1 = 100$. Given that $p = 1$, then how many individuals from this release cohort would we expect to see at occasion 2? Fairly easy calculation, if you remember that the apparent survival probability for residents is 0.8, and for transients is 0. Since 50 individuals in $\mathbf{R}_1$ are residents, then we expect that 80% of them (40/50) and be encountered at occasion 2. In contrast, of the 50 transient individuals in $\mathbf{R}_1$, all of them will emigrate, and thus no transients will be encountered at occasion 2. Now, what we would 'see' in our data is $\mathbf{R}_1 = 100$, and the number encountered at occasion 2 is 40. If we didn't know the starting, true proportions of residents and transients in the population, then our estimate of survival for the 100 marked and released individuals would be $(40/100) = 0.4$. Obviously, this is not a good estimate of true resident survival probability. It is biased low – we estimate an apparent survival probability of 0.4, but the true survival probability is 0.8! Clearly, our estimate is negatively biased.

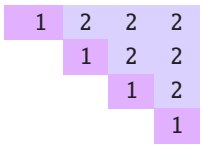
What about the interval between occasion 2, and occasion 3. Recall that at occasion 1, our release sample $\mathbf{R}_1$ consisted of both residents and transients. However, by occasion 2, our sample consists entirely of residents (since by definition the transients all emigrated during the interval between occasion 1 and 2). Thus, over the interval from occasion 2 to occasion 3, we would expect the estimated survival to reflect that for residents (i.e., $\varphi_R = 0.8$), since there are only residents released at occasion 2. And so on for the interval from occasion 3 to occasion 4, occasion 4 to occasion 5. . .

Now, what does this have to do with TSM models? Well, from the preceding, it is clear that during the interval after marking, the marked sample is a mix of residents and transients - let's call the apparent survival probability estimated over this interval φ_{M1}, where '*M1*' refers to the first year after marking - the *M* indicating 'marking'. Let φ_{M2+} be the survival rate estimated during the intervals after the first interval following marking, where '*M2+*' refers to years 2 and over after marking). Thus, for a single release cohort, we could write:

Now, suppose that we release **R** newly marked individuals at each occasion - if we have 5 total occasions, we could write



Which in **MARK** (PIM) format, is equivalent to

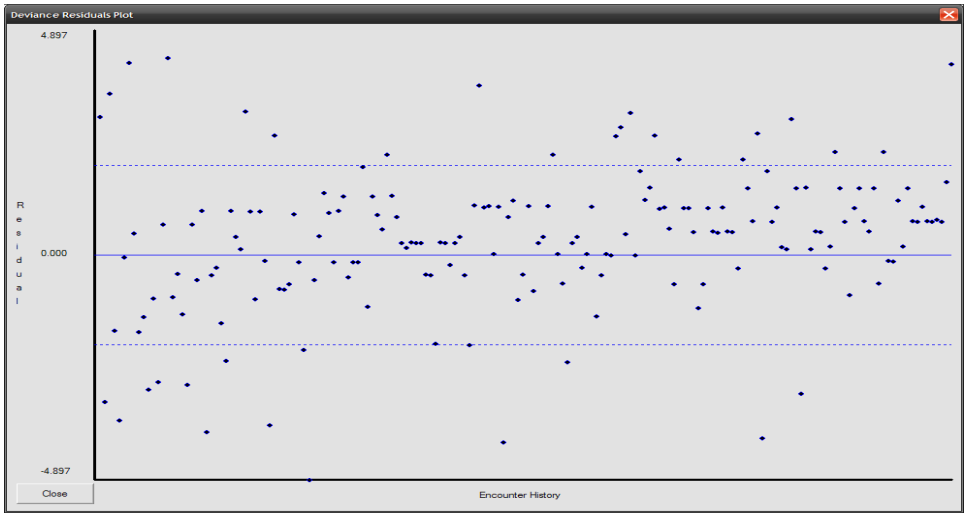


Look familiar? It should - it's identical to the PIM for a 2 age-class model, with no time variation in either age class! Here, though, we are **not** dealing with true 'age' differences, but an effect of heterogeneity in the sample. And yet, despite this difference, the PIM is identical to the PIM we might use for an 'age' model. And, therein lies the potential for confusion - if you use a PIM with 'age' structure to analyze a data set that doesn't have real age effects (for example, a data set where everyone is marked as an adult), you are bound to confuse some editors and reviewers who may not know that an 'age' model refers to a particular model structure, and not necessarily to true age. So, we recommend use of 'TSM models' as the generic name for such models - if you really are dealing with 'true' age effects (such as might be the case when you have individuals marked as young), then you probably should call them 'age models'. But otherwise, 'TSM models' may be less confusing for readers who don't see the subtle connections.

To reinforce the utility of TSM models, let's consider an example of a simulated data set which contains both resident and transient animals. We'll simulate 8 occasions. At each occasion, we'll release 500 newly marked individuals. To keep the example (and modeling) simple, we'll assume that at each occasion, the sample consists of the same proportion of residents to transients (70% residents, 30% transients). Assume that survival of residents is a constant, 0.8. Assume also that the encounter probability of residents (conditional on surviving) is 0.7. The data are contained in the file `transient.inp`.

Open up **MARK**. To reinforce the typical steps in an analysis, we'll start with a GOF test of a general model to these data. Now, you might have enough biological insight into this population that you make an a priori decision to fit a general model with structure which accounts for transients (the TSM models we're introducing here, for example). But, for the moment, let's 'pretend' we know nothing about these data - under such circumstances, we typically would start with a standard 'time-dependent' model. We'll do so here - we'll fit model $\{\varphi_t p_.\}$ to the data. Go ahead and run this model in **MARK**. If we run a median $\hat{c}$ GOF test, we find that the estimated $\hat{c}$ is ~ 1.9. While this is not unreasonably large (i.e., it does fall below our rule-of-thumb $\hat{c} \leq 3$), it does give us some indication of lack-of-fit of this model to the data. This might suggest that we need to find a better general starting model.

How else might we examine the lack-of-fit? Well, in **MARK** there is an option to allow you to examine the *deviance residuals* for any model in the model set. The deviance residual plot for model $\{\varphi_t p_.\}$ is shown at the top of the next page.



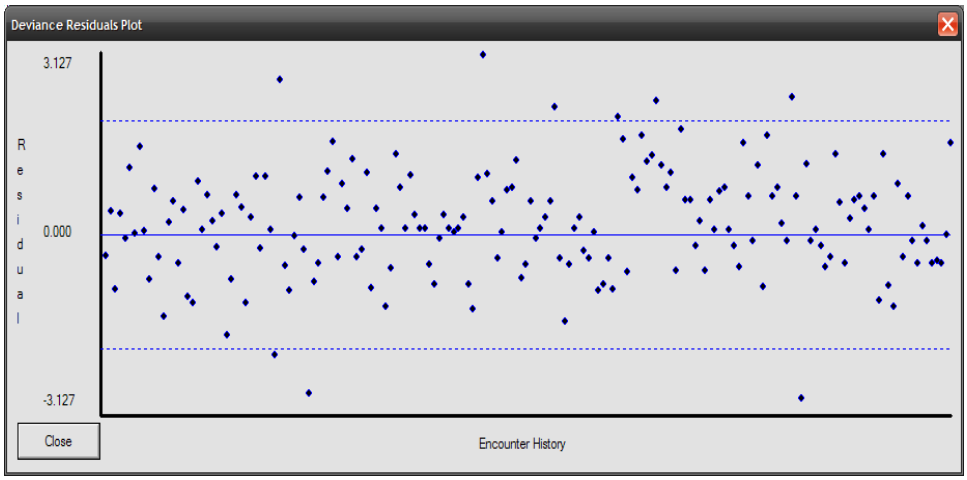
If the model was a 'good-fitting model', we would expect that there would be no 'trend' (non-randomness) in the pattern of the residuals - half of the residuals should be above the 0.000 line, and half should be below. Further, if there is no extra-binomial variation, then most of these residuals should be fairly close to the 0.0000 line (between the two horizontal dashed lines in the preceding figure). However, in this case, our estimate of $\hat{c}$ suggested pretty strongly that there is lack-of-fit of this model to the data. We see graphical support for this in the residual plot, since (i) there is a clear asymmetry of distribution of the points around the 0.000 line (in this case, more of the residuals are above the 0.000 than are below it), and (ii) a fair number of the residuals are well above/below the dashed lines. So, clearly, model $\{\varphi_t p_.\}$ is not a model that fits the data particularly well.

So, let's try fitting a TSM model - we 'suspect' this is likely to be a better model structure for these data (OK, we know it is, because these are simulated data). We'll fit the TSM model with 2 M-classes, constant over time in each class (call this model $\{\varphi_{M2-./.} p_.\}$). Here are the results:

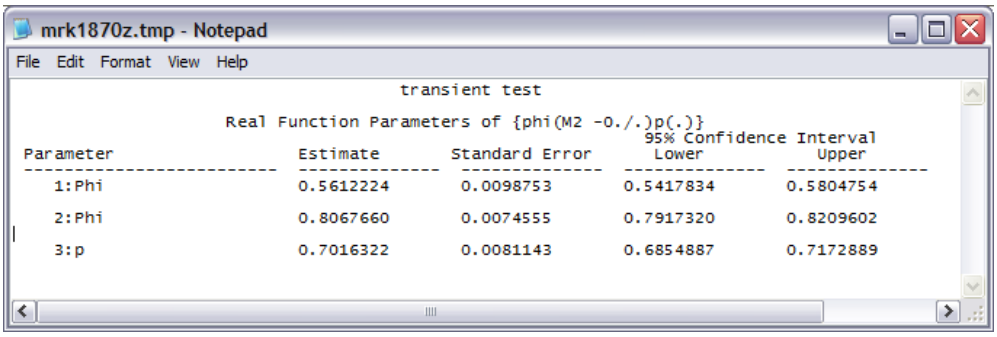
Results Browser: Live Recaptures (CJS)

Model	AICc	Delta AICc	AICc Weight	Model Likelihood	No. Par.	Deviance
{phi(M2 - ./.)p(.)}	12646.6257	0.0000	1.00000	1.0000	3	263.6537
{phi(t)p(.)}	12921.1302	274.5045	0.00000	0.0000	8	528.1391

Clearly, the model with 2 M-classes is the best model - it has 100% of the weight! Of course, this isn't surprising since it is the 'true' model for the simulated data. The point of note, however, is that a model with 'age' in the structure fits these data better than a simple time-dependent model, even though all the simulated individuals are of the same chronological age (i.e., even though there are no true age effects in the data). We can confirm this by looking at the estimate of $\hat{c}$ for this model (~ 1.01; estimation of $\hat{c}$ is discussed in the next section), and by looking at the residual plot (shown at the top of the next page). We see that in this case, the data are distributed evenly above/below the 0.000 line, indicative of pretty good fit of the model to the data.



If we look at the estimates from the best model



we see that the estimate of survival of the first *M*-class (i.e., in the interval following marking) is negatively biased, 0.5612 (as expected), whereas the estimate of survival for subsequent intervals, 0.8067, is very close to the 'true' underlying parameter value 0.8.

Recall that after the first interval, the marked sample for a given cohort consists entirely of residents, who will survive at the resident survival probability (0.8, in this case). The estimate of p is not influenced by transient individuals (since they are all out of the sample by the second occasion) - remember that p is estimated conditional on surviving and remaining in the sample. So, clearly, transient individuals do not influence the estimate of p.

So, by fitting a TSM model to these data, we were able to derive an unbiased estimate of resident survival probability - we accomplish this by accounting for differences in survival as a function of time since marking-in this case, time since marking corresponds to changes not in age, but in the heterogeneity of the marked individuals in the sample.

begin sidebar

proportion of transients in newly marked sample

For completeness, we note that from the estimates of survival for the 2 *M*-class, we can derive an estimate of the proportion of *residents* in the 'newly marked' sample of individuals for a given interval by taking $\hat{\varphi}_{T+M}$ (i.e., estimated survival over the a particular interval where *both* residents and transients are in the marked sample) and dividing it by $\hat{\varphi}_R$ (i.e., the estimate of survival over the same interval

when only residents are in the sample). In other words, the estimates along the diagonal divided by the off-diagonal estimates, for a given interval – i.e., within a column of the PIM. In this case, where the underlying probabilities and proportions of residents and transients in the newly marked sample were constant over time (since we simulated them that way), the proportion of *residents* would be given as $0.561/0.807 = 0.695 \sim 70\%$, which was the true value used in the simulation. Clearly, $1 - (0.561/0.807)$ gives the proportion of *transients* in the sample (*note*: in the sample, **not** necessarily in the population).

Why does this work? Simple algebra. Let the total number of birds released at occasion 1 for a given cohort be $(N_T + N_R)$, where N_T = number of transients in the sample, and N_R = number of residents in the sample. Since only residents will be encountered at occasion 2 (conditional on surviving), then the number of individuals expected alive at occasion 2 is $\varphi_R N_R$ (where φ_R is the survival probability of residents). Thus, the apparent survival probability over the first interval is $\varphi_R N_R/(N_R + N_T)$. From occasion 2 to occasion 3, within the same cohort, the population consists entirely of residents. The survival probability of residents is φ_R. Thus, the survival in the first interval divided by the survival after the first interval is $[\varphi_R N_R/(N_R + N_T)]/\varphi_R = N_R/(N_R + N_T)$, which is the proportion of residents in the sample! For details, see the Pradel *et al*. (1997) paper.

To get an estimate of the SE for the proportion of residents (or transients), we could use the Delta method (described in Appendix B).

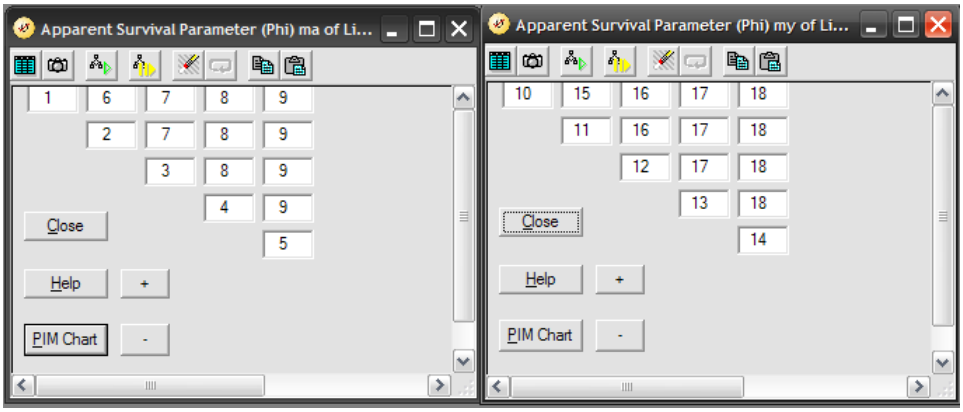
end sidebar

7.4.1. Age, transience and the DM – a complex example

We end this section with consideration of a more complex example. The purpose of this example is to reinforce our understanding of building design matrices for TSM models. Such models are arguably among the most complicated you are likely to work with in practice, and if you are able to work through this example, you can be fairly confident that you're likely to be able to handle even fairly complex problems.

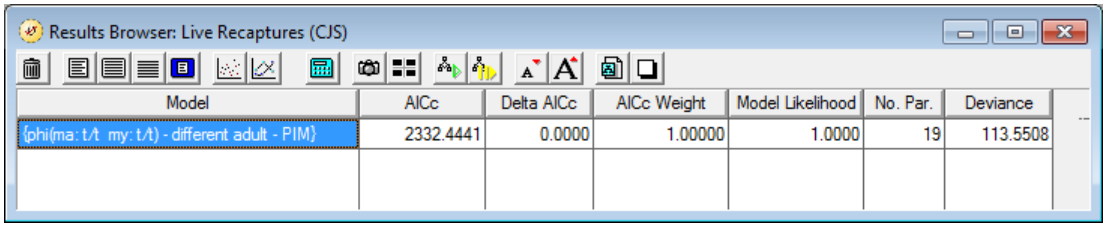
Here we imagine a 6 occasion study where where have individuals marked as young, and adults. Amongst those marked as young, we anticipate temporal variation in survival over the first year (i.e., within the first TSM class), but constant among individuals surviving to become adults. Among those marked as adults, we suspect that the first TSM class consists of some temporally variable mixture of resident and transient individuals (defined as in the preceding example). After the first year after marking (i.e., after the first TSM class), we anticipate adult survival is constant over time. However, we have some reason to suspect that survival of adults may differ as a function of the age at which individuals were initially marked and released. We expect that encounter probability is constant over time, with no TSM structure, and does not differ as a function of age at marking. The data for this example problem are contained in `age-transients-full.inp`. Note that in this input file, the encounter frequencies for the 'marked as adult' group come first, followed by the encounter frequencies for the 'marked as young' group).

Start a new project, remembering that you have 2 attribute groups, which we'll label '`ma`' and '`my`' (corresponding to 'marked as adult' and 'marked as young', respectively). Since this is a fairly complex problem, we'll start by constructing our general model using PIMs. For our general model, we'll assume full time dependence throughout. We'll label as $\{\varphi_{\texttt{ma}:t/t\ \texttt{my}:t/t,\Delta\texttt{adt}} p_{.}\}$, where `ma` refers to 'marked as adults', `my` refers to 'marked as young', and Δ`adt` refers to 'adult survival differing as a function of age of marking'. Clearly, coming up with a 'naming convention' for TSM models can prove somewhat of a challenge. The PIMs corresponding to apparent survival φ for 'marked as adults' and 'marked as young' are shown at the top of the next page.



For the 'marked as adults' group (left-hand PIM), parameters $1 \rightarrow 5$ correspond to the first TSM class (which contains a mixture of transients and residents), while parameters $6 \rightarrow 9$ correspond to 'adult' (ie., resident) survival. For the 'marked as young' group (right-hand PIM), parameters $10 \rightarrow 14$ correspond to the first TSM class (i.e., age $0 \rightarrow 1$), while parameters $15 \rightarrow 18$ correspond to the adults. With a constant and equal encounter probability for both groups, we have 19 total parameters in our model.

Run this model, and add the results to the browser. Note that we add the word 'PIM' to the model name to indicate that this version of the general model was constructed using PIMs. We will use this model to 'confirm' that the we have constructed the equivalent DM-based model correctly.



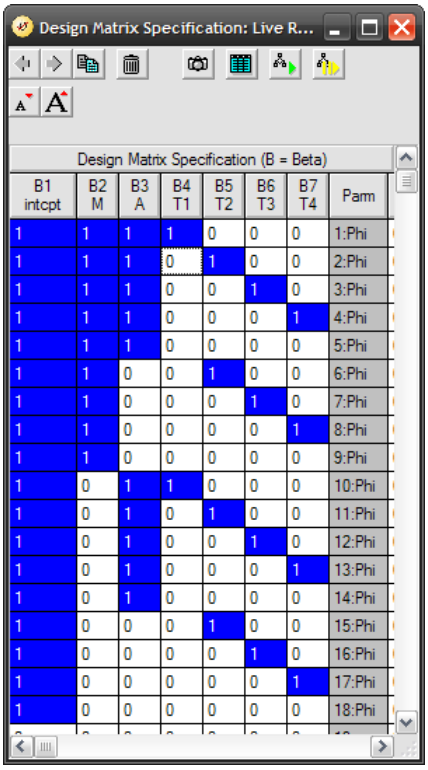
Let's start the construction of the DM corresponding to the parameter structure specified by these PIMs by writing out the first few terms in our linear model, corresponding to the 'main effects':

$$\varphi = \texttt{inctp} + \texttt{M} + \texttt{A} + \texttt{T}$$

where `M` refers to the 'age of marking', `A` refers to 'age' (or TSM class) within age of marking group), and `T` refers to time. Since there are 2 age of marking groups, then we need one column in the DM to code for `M`. Since there are 2 age (TSM) classes within each age of marking group, we need one column in the DM to code for `A`. And finally, since there are 6 occasions, there are 5 intervals, so we need four columns in the DM to code for `T`. So, our linear model so far is

$$\begin{aligned}\varphi &= \texttt{inctp} + \texttt{M} + \texttt{A} + \texttt{T} \\ &= \beta_1 + \beta_2(M) + \beta_3(A) + \beta_4(T_1) + \beta_5(T_2) + \beta_6(T_3) + \beta_7(T_4)\end{aligned}$$

Let's start our DM by coding these 8 terms (shown at the top of the next page). After the single column for the intercept, we code the `M` effect: we use `1`'s to indicate 'marked as adult', and `0`'s to indicate 'marked as young'. This is followed by the column coding for the `A` effect – here we use `1`'s to code for the first



B1 intcpt	B2 M	B3 A	B4 T1	B5 T2	B6 T3	B7 T4	Parm
1	1	1	1	0	0	0	1:Phi
1	1	1	0	1	0	0	2:Phi
1	1	1	0	0	1	0	3:Phi
1	1	1	0	0	0	1	4:Phi
1	1	1	0	0	0	0	5:Phi
1	1	0	0	1	0	0	6:Phi
1	1	0	0	0	1	0	7:Phi
1	1	0	0	0	0	1	8:Phi
1	1	0	0	0	0	0	9:Phi
1	0	1	1	0	0	0	10:Phi
1	0	1	0	1	0	0	11:Phi
1	0	1	0	0	1	0	12:Phi
1	0	1	0	0	0	1	13:Phi
1	0	1	0	0	0	0	14:Phi
1	0	0	0	1	0	0	15:Phi
1	0	0	0	0	1	0	16:Phi
1	0	0	0	0	0	1	17:Phi
1	0	0	0	0	0	0	18:Phi

TSM class within each age of marking group, and `0`'s to code for the second TSM class. Recall that for 'marked as adults' the first TSM class reflects a mixture of residents and transients, whereas for the 'marked as young' group, the first TSM class are juveniles. Finally, 4 columns to code for each of the time intervals (here we use the final time interval as the reference interval). Note again that the second TSM class is not represented in the first time interval (and thus the time coding for the second TSM class begins with the second interval).

Now, for the interactions of the main effects. We know from earlier in the chapter that in general, it is the construction of the various interaction terms which poses the largest challenge. The key is in remembering that coded interactions must be 'plausible'. So far, we have 7 parameters for φ in our DM, meaning, we have $(18 - 7) = 11$ parameters remaining for the interaction terms. Which in turn means that if we have more or less than 11 columns for our interactions, we've made a mistake somewhere.

What are the 'possible' interactions (which may or may not be the same as those which are 'plausible'). For a model with 3 main effects (`M`, `A` and `T`), there are clearly 3 possible two-way interactions (`M.A`, `M.T`, and `A.T`), and 1 possible three-way interaction (`M.A.T`). Remember, that if any of the two-way interactions aren't plausible, then neither is the three-way interaction (since the three-way interaction consists of the product of a given two-way interaction and the remaining effect). So, in theory, our linear model could (in theory) look like

$$\begin{aligned}\varphi = \texttt{INTCPT} &+ \texttt{M} + \texttt{A} + \texttt{T} \\ &+ \texttt{M.A} + \texttt{M.T} + \texttt{A.T} \\ &+ \texttt{M.A.T}\end{aligned}$$

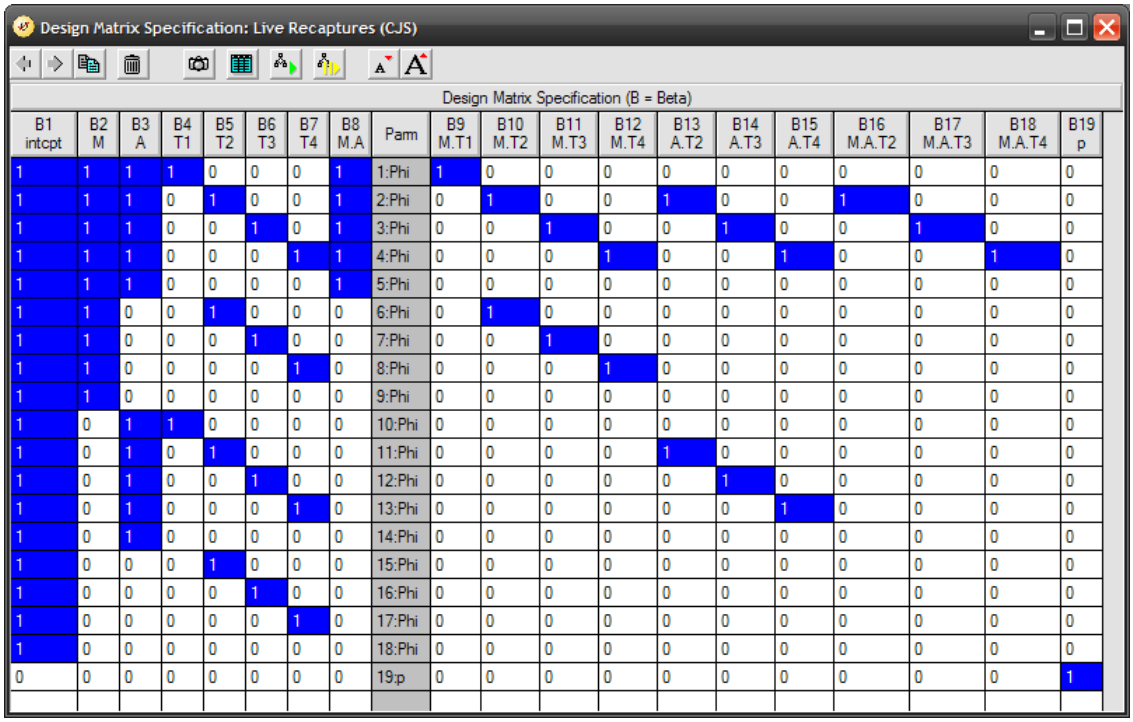
So, which interactions are plausible? Start with the `M.A` interaction. We see from the DM that there is

'symmetry' between the two groups ('marked as adult' and 'marked as young') in the TSM structure. So, there is a plausible `M.A` interaction term. Since the `M.A` interaction is the product of two single columns, then clearly the `M.A` interaction is coded as a single column. So, that would give us (7+1)= 8 columns; 10 remaining.

What about the `M.T` interaction? Again, we see that the time columns are symmetrical between the two age of marking groups. In other words, the time coding for 'marked as adults' is perfectly replicated for the 'marked as young' group. So, we have a full `M.T` interaction (1 column for `M` with 4 columns for time, yield 4 columns for the `M.T` interaction term). So, (8+4)=12 columns coded so far; 8 left.

Next, the `A.T` interaction. Here, we simply need to remember that an `A.T` interaction is not plausible for all time steps. Since there are no individuals representing the second TSM class in the first time interval for either the 'marked as adult' or 'marked as young' groups, then `A.T` interactions are plausible only for time intervals 2 → 5). So, 3 columns for the `A.T` interaction. So, to this point, (12+3)=15 columns; so, 3 left.

Finally, the three-way interaction `M.A.T`. All we need to do at this point is take the `A.T` interaction, and multiply through by the `M` term. We know that we have 3 columns left to code for the three-way interaction. Fortunately 1 (for `M`) times 3 (for the `A.T` interaction columns) = 3! Nice when things 'add up'. Here is the completed design matrix for our general model – pay particular attention to the column labels.



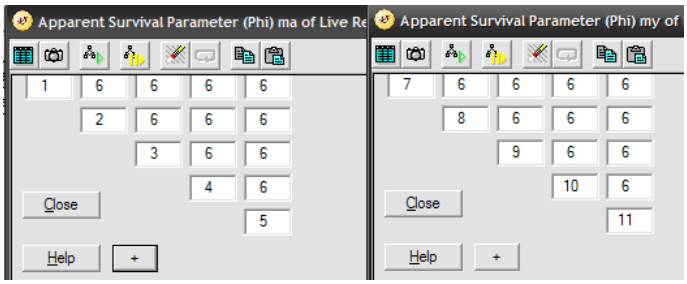
B1 intcpt	B2 M	B3 A	B4 T1	B5 T2	B6 T3	B7 T4	B8 M.A	Parm	B9 M.T1	B10 M.T2	B11 M.T3	B12 M.T4	B13 A.T2	B14 A.T3	B15 A.T4	B16 M.A.T2	B17 M.A.T3	B18 M.A.T4	B19 p
1	1	1	1	0	0	0	1	1:Phi	1	0	0	0	0	0	0	0	0	0	0
1	1	1	0	1	0	0	1	2:Phi	0	1	0	0	1	0	0	1	0	0	0
1	1	1	0	0	1	0	1	3:Phi	0	0	1	0	0	1	0	0	1	0	0
1	1	1	0	0	0	1	1	4:Phi	0	0	0	1	0	0	1	0	0	1	0
1	1	1	0	0	0	0	1	5:Phi	0	0	0	0	0	0	0	0	0	0	0
1	1	0	0	1	0	0	0	6:Phi	0	1	0	0	0	0	0	0	0	0	0
1	1	0	0	0	1	0	0	7:Phi	0	0	1	0	0	0	0	0	0	0	0
1	1	0	0	0	0	1	0	8:Phi	0	0	0	1	0	0	0	0	0	0	0
1	1	0	0	0	0	0	0	9:Phi	0	0	0	0	0	0	0	0	0	0	0
1	0	1	1	0	0	0	0	10:Phi	0	0	0	0	0	0	0	0	0	0	0
1	0	1	0	1	0	0	0	11:Phi	0	0	0	0	1	0	0	0	0	0	0
1	0	1	0	0	1	0	0	12:Phi	0	0	0	0	0	1	0	0	0	0	0
1	0	1	0	0	0	1	0	13:Phi	0	0	0	0	0	0	1	0	0	0	0
1	0	1	0	0	0	0	0	14:Phi	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	15:Phi	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	1	0	0	16:Phi	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	1	0	17:Phi	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	18:Phi	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	19:p	0	0	0	0	0	0	0	0	0	0	1

To confirm that we've have correctly constructed the DM, run the model, making sure to change the model name to indicate that we're using the DM, and add the results to the browser (shown at the top of the next page). Notice that the model deviances between the model built with PIMs, and the one we just fit using the DM approach, are identical. This indicates that our DM is correct.

Results Browser: Live Recaptures (CJS)

Model	AICc	Delta AICc	AICc Weight	Model Likelihood	No. Par.	Deviance
{ma: t/t my: t/t - different adult - PIM}	2332.4441	0.0000	0.50000	1.0000	19	113.5508
{ma: t/t my: t/t - different adult - DM}	2332.4441	0.0000	0.50000	1.0000	19	113.5508

A final test of our understanding. Suppose we want to build a reduced parameter model, where survival varies temporally for the first TSM class for both 'marked as adult' and 'marked as young' groups, but with no time variation in the second TSM class, and the same value between marking groups (in other words, time-invariant survival of the second TSM class that does not vary as a function of the age of marking). The PIM structure for this model would look something like



But, recall that we'd like to build these models by modifying the DM of our general model. Here is the modified DM which corresponds to this reduced parameter model

Design Matrix Specification: Live Recaptures (CJS)

Design Matrix Specification (B = Beta)

B1: Int	B2: M	B3: A	B4: T1	B5: T2	B6: T3	Parm	B7: T4	B8: M.T1	B9: M.T2	B10: M.T3	B11: M.T4	B12: p
1	1	1	1	0	0	1:Phi	0	1	0	0	0	0
1	1	1	0	1	0	2:Phi	0	0	1	0	0	0
1	1	1	0	0	1	3:Phi	0	0	0	1	0	0
1	1	1	0	0	0	4:Phi	1	0	0	0	1	0
1	1	1	0	0	0	5:Phi	0	0	0	0	0	0
1	0	0	0	0	0	6:Phi	0	0	0	0	0	0
1	0	0	0	0	0	7:Phi	0	0	0	0	0	0
1	0	0	0	0	0	8:Phi	0	0	0	0	0	0
1	0	0	0	0	0	9:Phi	0	0	0	0	0	0
1	0	1	1	0	0	10:Phi	0	0	0	0	0	0
1	0	1	0	1	0	11:Phi	0	0	0	0	0	0
1	0	1	0	0	1	12:Phi	0	0	0	0	0	0
1	0	1	0	0	0	13:Phi	1	0	0	0	0	0
1	0	1	0	0	0	14:Phi	0	0	0	0	0	0
1	0	0	0	0	0	15:Phi	0	0	0	0	0	0
1	0	0	0	0	0	16:Phi	0	0	0	0	0	0
1	0	0	0	0	0	17:Phi	0	0	0	0	0	0
1	0	0	0	0	0	18:Phi	0	0	0	0	0	0
0	0	0	0	0	0	19:p	0	0	0	0	0	1

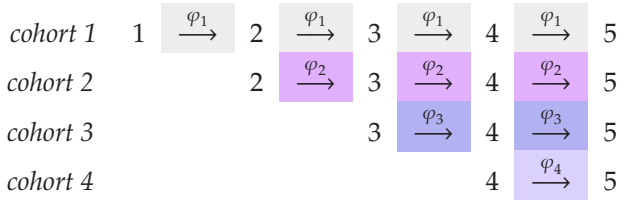
Pay particular attention to the way we've modified the `M` coding: since we've eliminated any effect of age of marking on survival for the second TSM class, we eliminate the *1*'s coding for the second TSM class for the 'marked as adults' group. The rest of the DM should be fairly self-explanatory.

7.5. Age/TSM models and GOF

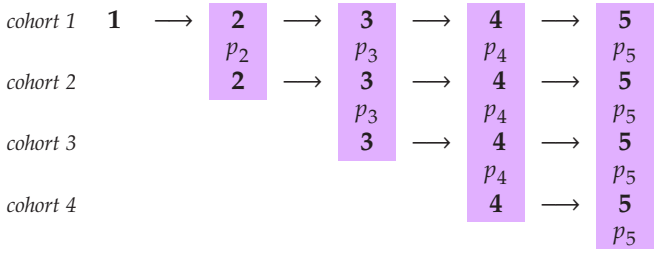
If you remember back to Chapter 5, you may recall that there are various ways to test for GOF for a given model. Looking at deviance residual plots was described above. Note that because age/TSM models have more parameters than standard time-dependent CJS models, then in fact age/TSM models are more 'general' than CJS models, and as such, should be models for which you assess fit (if such models are included in your candidate model set). You can use either **RELEASE** (for model with 2 'classes', by adding some of the individual component tests – discussed in Chapter 5), the bootstrap or median-$\hat{c}$ methods, or (and perhaps most usefully), program **U-CARE**, which has a large number of component tests specific for age/TSM models (including transience testing, for example).

7.6. Cohort models

The time-dependent CJS model assumes that neither survival nor recapture differ among release cohorts. Is this a reasonable assumption? Suppose, for example, that animals newly marked on occasion 3 were present in the population on occasion 1, but were simply 'missed' from the marking sample. Perhaps there is a 'reason' why these animals were missed - and this reason might influence subsequent survival or (perhaps more likely) recapture probability. Another common example is that cohorts differ in the environmental conditions experienced by the individuals during marking, such that subsequent survival, or recapture, or both, are affected. In essence then, a cohort model is simply one where survival and/or recapture probabilities differ as a function of the cohort an animal is first released into. In its simplest form, a cohort model for survival (but **not** recapture) can be represented as:

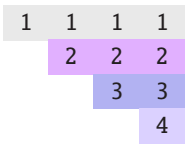


For recapture, we still maintain the usual time-dependent parameter structure:

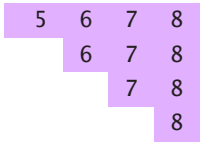


In this case, survival is constant over time, but differs among cohorts. How would this be represented by **MARK**? In other words, what would the PIMs look like?

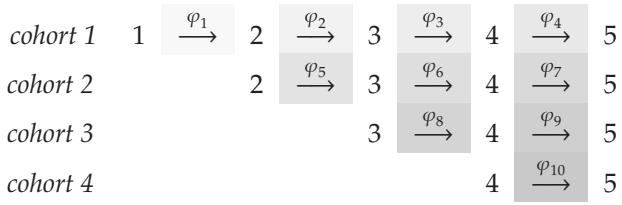
For survival,



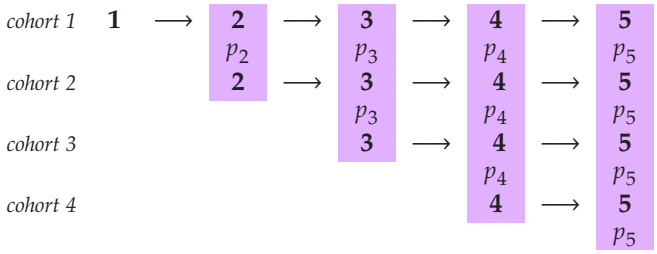
and for recapture



Of course, we could add time-dependence within-cohort, but this will substantially increase the number of parameters. For example, consider the following (cohort×time) model for survival (shown at the top of the next page) – estimates differ over time *within* cohort, but over a given interval, they differ *among* cohorts.



Not only is the number of parameters increased significantly, but the number of estimable and non-estimable parameters is also changed. For example, suppose that the preceding model is applied to survival, but simple time-dependence for recaptures.



How many estimable parameters would there be? As discussed in Chapter 4, one way to count the number of estimable parameters for any single-group model is to write out the probability statements for each of the 'saturated' capture histories.

For this example, the saturated histories, and their corresponding probability statements are:

encounter history	*probability*
11111	$\varphi_1 p_2 \varphi_2 p_3 \varphi_3 p_4 \varphi_4 p_5$
01111	$\varphi_5 p_3 \varphi_6 p_4 \varphi_7 p_5$
00111	$\varphi_8 p_4 \varphi_9 p_5$
00011	$\varphi_{10} p_5$

How many unique, identifiable parameters are there? Clearly, the key to answering this question lies in the terminal product terms (i.e., $\varphi_4 p_5$, $\varphi_7 p_5 \ldots$). Each of these β terms contains p_5. Is p_5 estimable? No - not without more information! As such, we have 4 different β terms - and thus 13 parameters.

As you might imagine, with progressively complex models, it can be more work to count the number of parameters. And, clearly, knowing the number of parameters is essential for model selection.

begin sidebar

cohort models with individuals marked as young

What about cohort models where individuals are marked as young? Well, in such cases, you run into an interesting consideration: for animals marked as young, as time passes, they get older. Thus, within a cohort, age and time are the same (as we've discussed previously). In fact, age, time and cohort are collinear, since

age = time - cohort

As such, you **can't** consider all three factors simultaneously. You need to 'collapse' over at least 1 of the 3 dimensions (i.e., you can look at age and cohort, or age and time, or cohort and time, but not all 3).

end sidebar

7.6.1. Building cohort models: PIMS and design matrices

The preceding suggests pretty strongly that in most cases, you'll build cohort models using PIMs. But, as you by now no doubt realize, if you want to apply any constraints to the model, or build additive models, then you'll need to use the design matrix.

Although the basic ideas behind building design matrices for cohort models are pretty much the same as we've seen before, you need to think a bit when it comes to (`cohort` × `time`) models. Let's have a quick look at a couple of them.

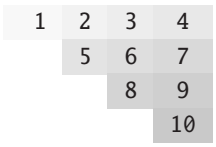
Suppose you have a 5 occasion mark-recapture study, and you are interested in building a series of cohort models for survival. Suppose your candidate model set consists of

$$\{\varphi_{cohort}\}$$
$$\{\varphi_{cohort.time}\}$$

The basic PIM structure for φ_{cohort} (i.e., cohort effects only, no time variation) is

1	1	1	1
	2	2	2
		3	3
			4

whereas the PIM structure for $\varphi_{cohort.time}$ is



Now, there are a couple of equivalent design matrices for both of these models. Recall from Chapter 6 that we can often use the identity matrix to code for certain linear models in place of the intercept-based approach. The advantage of the latter, however, is that it allows for easy testing of certain constrained models.

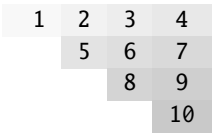
For the first model φ_{cohort}, the two equivalent design matrices would be

$$\mathbf{X}_1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \mathbf{X}_2 = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

The first design matrix ($\mathbf{X}_1$) is the standard identity matrix. In the second design matrix ($\mathbf{X}_2$), the first column is the intercept, while the next 3 columns code for the level of the 'treatment', cohort. Since there are 4 cohorts in a 5 occasion study, then we need 3 columns. Each row of the matrix corresponds to a different cohort.

OK, probably pretty straightforward, at this point. What about model $\{\varphi_{cohort.time}\}$? Well, as it turns out, this is somewhat more complicated, for some of the same reasons building the design matrix for the fully time-dependent age model was a few pages back: because there are some 'cohort × time' interactions which do not occur. As such, it turns out there are several equivalent intercept-based linear models that will yield the same fit. We'll spend just a moment here on this - if for no other reason that it forces us to think a bit more deeply about design matrices.

Now, for model $\varphi_{cohort.time}$, the (10×10) identity matrix design structure corresponding to the PIM would look like:



$$\mathbf{D} = \begin{bmatrix} d_{1,1} & 0 & \dots & d_{1,10} \\ 0 & d_{2,2} & \dots & d_{2,10} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & d_{10,10} \end{bmatrix} = \begin{bmatrix} 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 1 \end{bmatrix}$$

What about the intercept-based design matrix? Obviously, we need a column for the `intercept`. The `cohort` and `time` columns should be pretty straightforward: we have 4 cohorts, so 3 columns: let '`1 0 0`' be the coding for cohort 1, '`0 1 0`' be the coding for cohort 2, '`0 0 1`' be the coding for cohort 3, and '`0 0 0`' be the coding for cohort 4. But, recall that the number of time intervals (parameters) for each cohort decrements by one with each successive cohort (i.e., for cohort 1 there are 4 intervals, for cohort 2 there are 3 intervals, for cohort 2 there are 2 intervals, and for cohort 4, there is only the final interval).

So, thus far, our design matrix looks like

$$\begin{bmatrix} 1 & 1 & 0 & 0 & \dots \\ 1 & 1 & 0 & 0 & \dots \\ 1 & 1 & 0 & 0 & \dots \\ 1 & 1 & 0 & 0 & \dots \\ 1 & 0 & 1 & 0 & \dots \\ 1 & 0 & 1 & 0 & \dots \\ 1 & 0 & 1 & 0 & \dots \\ 1 & 0 & 0 & 1 & \dots \\ 1 & 0 & 0 & 1 & \dots \\ 1 & 0 & 0 & 0 & \dots \end{bmatrix}$$

Now, for time, the approach is similar: we have 5 occasions (4 intervals), so therefore, we need 3 columns. We let '1 0 0' be the first time interval, '0 1 0' be the second time interval, '0 0 1' be third interval, and '0 0 0' the final time interval. This is the usual 'reference cell' coding approach we discussed at length in Chapter 6. Adding these time columns to the design matrix gives us

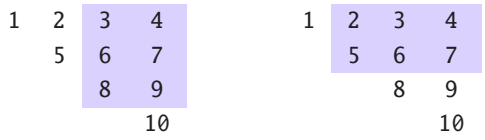
$$\begin{bmatrix} 1 & 1\ 0\ 0 & 1\ 0\ 0 & \dots \\ 1 & 1\ 0\ 0 & 0\ 1\ 0 & \dots \\ 1 & 1\ 0\ 0 & 0\ 0\ 1 & \dots \\ 1 & 1\ 0\ 0 & 0\ 0\ 0 & \dots \\ 1 & 0\ 1\ 0 & 0\ 1\ 0 & \dots \\ 1 & 0\ 1\ 0 & 0\ 0\ 1 & \dots \\ 1 & 0\ 1\ 0 & 0\ 0\ 0 & \dots \\ 1 & 0\ 0\ 1 & 0\ 0\ 1 & \dots \\ 1 & 0\ 0\ 1 & 0\ 0\ 0 & \dots \\ 1 & 0\ 0\ 0 & 0\ 0\ 0 & \dots \end{bmatrix}$$

How big does the design matrix need to be? Well, there are 10 parameters, so the full $\varphi_{cohort*time}$ model corresponds to a (10×10) design matrix. Thus, we have 3 more columns to complete. OK, now the challenge. How can we code an interaction of (cohort × time) in only 3 columns, if there are 4 cohorts! Normally, we think of the interaction term(s) as products of the main factors in the linear model: in this case, the product of the 3 columns for 'cohort', and the 3 columns for 'time', which would yield 9 columns, not 3! But, if you look closely at the PIM (below), you'll see that not all interactions are possible. This is exactly the same issue we faced earlier when we were looking at interactions in models for analysis of data of individuals marked as both young, and adults – not all interactions were possible. Here again is the PIM for model $\varphi_{cohort*time}$

1	2	3	4
	5	6	7
		8	9
			10

We see that for cohort 1, time interval 1 does not interact with any other cohort. We also see that cohorts 1 and 2 interact in both time interval 2, interval 3, and time interval 4. We see that cohort 3 only interacts with any other cohort (cohorts 1 and 2) in time intervals 3 and 4. For cohort 4, there appears to be an interaction in time interval 4, but, parameter 10 is not identifiable, so in fact, it doesn't really

interact with anything! So, we really have only 2 interaction blocks (indicated in the shaded 'blocks' in the following):



As it turns out, you only need to code either of these 'interaction blocks' to get the appropriate model fit (and parameter estimates). So, for example, for the second of these two 'interaction blocks':

$$\begin{bmatrix}
1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\
1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\
1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\
1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\
1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\
1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}$$

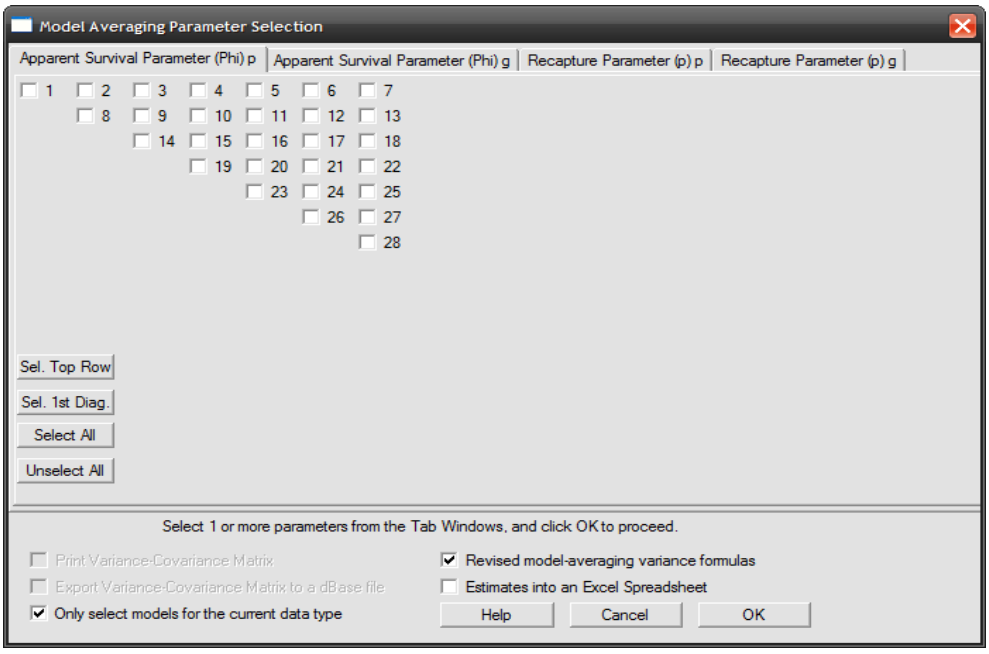
Try it and see! A bit tricky at first, but hopefully you'll see the basic logic. Again, full (`cohort` × `time`) models are relatively rare, owing to significant identifiability problems, but constrained models, especially where constraints are applied within cohort, are not uncommon. And to do this, you need to be able to build the design matrix for the full (`cohort` × `time`) model(s) first.

7.7. Model averaging and age/cohort models

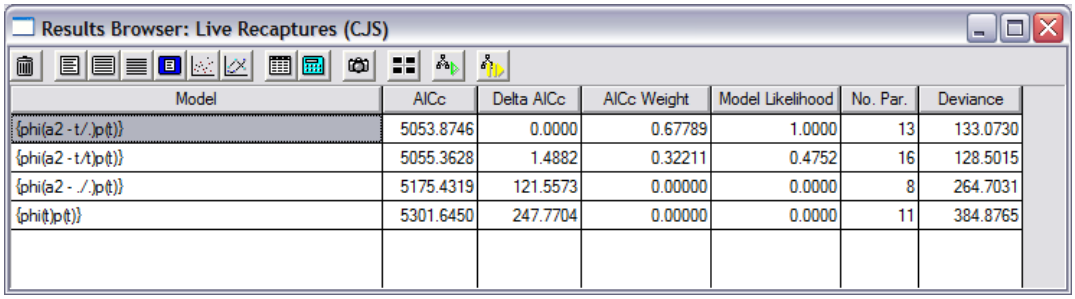
Back in chapter 4 we introduced the concepts of 'normalized AIC weights' and 'model averaging'. We considered the basic theoretical background, and introduced the mechanics of how to derive 'model averaged' parameter estimates in **MARK**.

However, as you may recall, there were a few parts of the description of the 'mechanics' of handling model averaging in **MARK** that were somewhat 'obtuse' (to be kind). That's because back in chapter 4 we had to make vague references to these things called 'age models' and 'cohort' models. New stuff in chapter 4, but by this point you're expert in both concepts, so we can re-visit some of the 'model averaging mechanics' and clear up a few points.

First, recall that for the swift data set, the model averaging window looked like the image shown at the top of the next page. For that data set, there were 2 groups (good colony, poor colony), and two parameters (survival probability, and recapture probability). Thus, 4 tabs along the top of the model averaging window (one for each group and parameter combination). In addition, for the swift data set, there were 8 occasions, so 7 intervals for survival. Now, if we look carefully at the 'triangular matrix' structure that the model averaging window presents us with, we see something that at this point should be a bit more understandable. What would a full (time × cohort) PIM look like for a data set with 8 occasions? Answer: just like the triangular matrix pictured at the top of the next page!

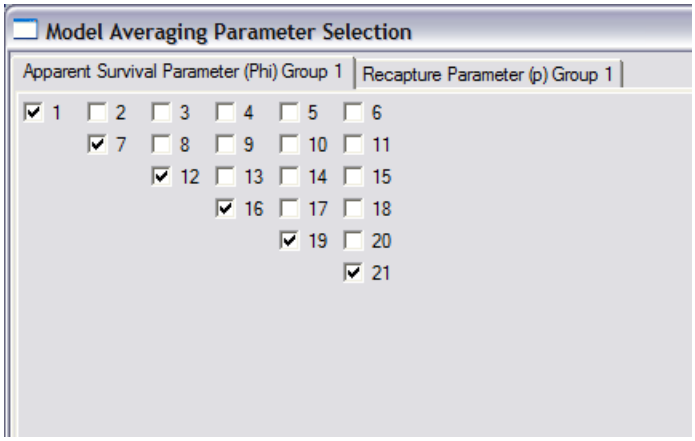


To make sure you understand model averaging, let's revisit an analysis we did earlier in this chapter, using some 'age data'. Recall that for these simulated data, consisting of 7 occasions (6 intervals), we had time-dependence in survival for the first age-class, and constant survival for the 'adult' age class. Here again were the results from our model fitting.



Model	AICc	Delta AICc	AICc Weight	Model Likelihood	No. Par.	Deviance
{phi(a2 -t/.)p(t)}	5053.8746	0.0000	0.67789	1.0000	13	133.0730
{phi(a2 -t/t)p(t)}	5055.3628	1.4882	0.32211	0.4752	16	128.5015
{phi(a2 - ./.)p(t)}	5175.4319	121.5573	0.00000	0.0000	8	264.7031
{phi(t)p(t)}	5301.6450	247.7704	0.00000	0.0000	11	384.8765

Again, we see that the most parsimonious model (model $\varphi_{a2-t/.}p_t$) is approximately twice as well supported as the next best model (0.67787/0.32213 = 2.1). What are the model averaged survival values for each interval for the first age-class? Again, remember that the first age-class is one-year in duration. To derive the model averaged survival probabilities for the first age-class, select '**Model Averaging**' from the '**Output**' menu in **MARK**. Note that this time, the model averaging window has only 2 tabs: for survival and recapture, respectively. The triangular matrix (which you now recognize as the corresponding (time × cohort) PIM structure) that the model averaging window presents is indexed from 1 to 21. The only thing you need to do at this stage is check the elements of this matrix corresponding to the first-year survival probabilities. What would they be? Yes – the elements along the diagonal: $\{1, 7, 12, 16, 19, 21\}$.



These correspond to the diagonal elements of the following PIM structure:

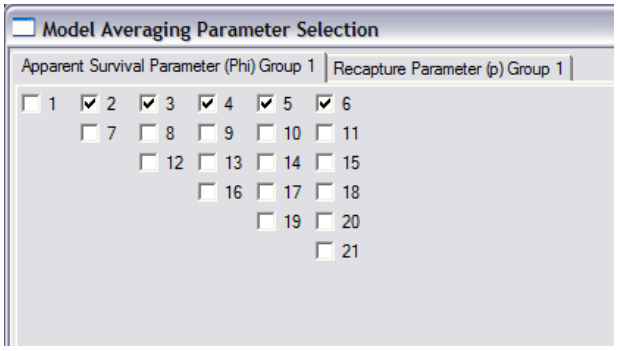
Once you've satisfied yourself that this is correct - and makes sense-then click the '**OK**' button. The model averaged survival values are:

	interval					
	1	**2**	**3**	**4**	**5**	**6**
estimate	0.2028	0.5711	0.5397	0.2890	0.3000	0.2802
SE	0.028	0.035	0.032	0.027	0.028	0.035

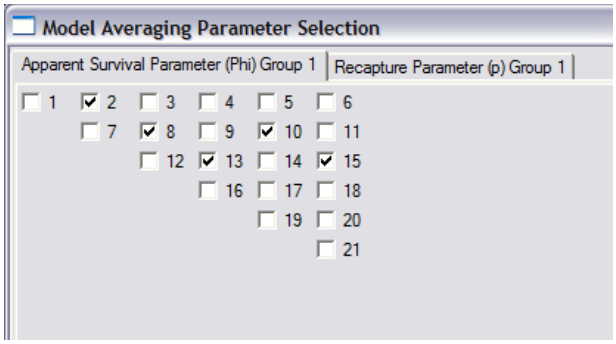
What about 'adult' survival? Again, the only trick is to remember that adult survival is 'above the diagonal' (look at the PIM schematic above; the adult 'elements' are the shaded elements above the diagonal). So, you simply need to check the appropriate elements of the triangular PIM structure in the model averaging window.

But (trick question) do you need to check all of them (i.e., 2 to 6, 8 to 11, 13 to 15, 17 to 18 and 20)? If we remind you that our models have time-structure only, but no cohort structure, you should realize with a bit of thought that the answer is 'no'. You simply need to check at least one of the 'above diagonal' elements in each column (each column where there is an above-diagonal element).

Thus, the following two model averaging windows shown on the next page will yield entirely equivalent results. Try it and see! But, more importantly, think about why they are equivalent.



or



Model averaging is an important technique, so make sure you really understand what's going on here. The model averaging window also forces you to come to grips with your understanding of age, cohort, and time models (remember, age = cohort-time). So, make sure you do 'get it'.

7.8. Summary

That is the end of Chapter 7. We have considerably expanded the range of 'underlying' models we can fit to mark-recapture data, by developing age/TSM and cohort models. We have also seen how constraints can be applied to these models as easily as we did with CJS models. We also revisited the idea of model averaging. These models are **very** widely used (the basic structure shows up various ways for a number of data types), so make sure you thoroughly understand the material in this chapter. In the next chapter we look at a very different data types 'dead recoveries'.

CHAPTER 8

'Dead' recovery models

The first chapters in this book focussed exclusively on live encounter 'mark-recapture' models, where the probability of an individual being seen on a particular sampling occasion was determined by 2 parameters: the probability the animal survived and the probability that an animal alive in state r at time i is alive and in state s at time i+1.

In this chapter, we move in a new direction altogether. We recall that 'classic' mark-recapture focuses on the problem of differentiating between (i) not seeing an animal because it is 'dead' (or permanently emigrated from the sample area) and (ii) simply 'missing' it, even though it is alive and in the sample area. In contrast, with 'dead recovery' analysis we are dealing with animals known to be dead (because they are recovered, generally in the process of harvest).

Echoing the seminal text by Brownie *et al*. (1985), it is sufficiently important to clearly distinguish between these two broad classes of sampling method (recovery and recapture) that we'll take a moment to elaborate on them. In the case of a recapture analysis, a single marked individual is potentially available for 'multiple encounters' – i.e., the individual may be 'seen' or 'recaptured' on more than one occasion. If you've worked through the preceding chapters of this book, this is entirely obvious to you. In contrast, in a recovery analysis, data are available on only a single, terminal 'encounter' (generally, the recovery event). Unlike recapture data, recovery data are treated as independent, mutually exclusive outcomes (i.e., a marked individual could be recovered in year 1, year 2,. . .or not at all during the duration of the study). While this is a clear difference, in fact, close examination shows deep similarity between the two models. The distribution of 'dead recoveries' reflects the realization of a series of probabilistic events. Just as each live encounter in a live encounter history reflects the underlying survival and recapture processes, so too does the distribution of 'dead recoveries'.

8.1. 'Brownie' parameterization

Consider the following example. An individual of a harvested species is marked. It can then experience one of 3 fates: (1) it can survive the year with some probability, (2) it can be 'harvested' (i.e., some 'action' leading to permanent removal) with some probability, or (3) it can 'die' from 'natural' causes (i.e., it might actually die from some reason other than harvest, or permanently emigrate the sampling area, at which point it appears dead. More on what constitutes the 'sampling area' for a dead recovery analysis in Chapter 9).

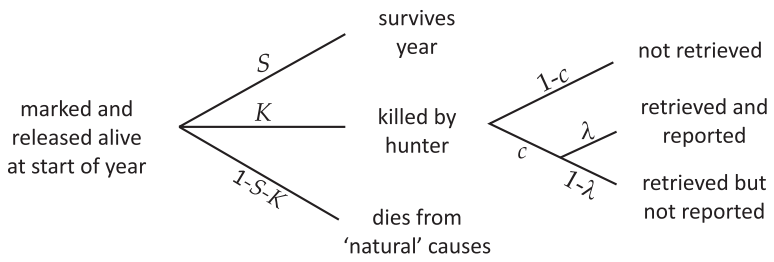
However, before this individual becomes 'dead recovery data', something else needs to happen – the 'harvest' needs to be 'reported'. This event reflects several underlying probabilistic events. Suppose

© Cooch & White (2014)

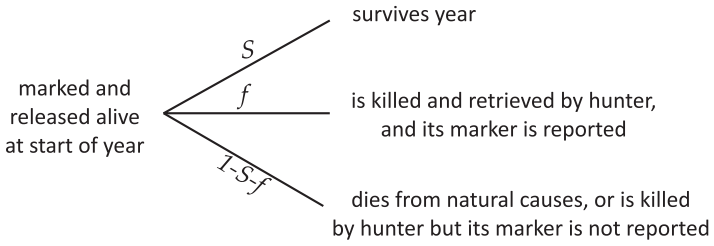
you're a waterfowl hunter, and you shoot a bird from your blind (or 'hide' for much of the world). This in itself does not constitute a recovery, since simply shooting the bird does not give us the information on who it is (i.e., its identification number). For this to happen, minimally (in most cases), the marked bird needs to be retrieved (i.e., physically handled, typically). Of course, there is some chance it won't be retrieved. If it is retrieved, however, then it might be 'reported' (i.e., the identification number submitted to some monitoring agency), or not. We'll let S = the probability that the individual survives the year. We separate sources of mortality into 'hunter' and 'natural'. The probability that the individual dies from either source is simply $1 - S$. The probability that it dies due to hunting is K. Thus, the probability that it dies from natural causes is $(1 - S - K)$.

Now for the only real complication (that is simple enough in principle, but has several interesting implications we will discuss later in this chapter). Conditional on being shot (i.e., killed by hunting, with probability K), then one of 3 things can happen. The individual may not be retrieved (a fairly common occurrence with some types of harvest - individuals are in fact killed by harvest, but the dead animal is not physically retrieved). The probability of being retrieved is c - thus, the probability of not being retrieved is $(1 - c)$. Conditional upon being retrieved, the hunter can either report the identification number (with probability λ), or not report the identification number (with probability $1 - \lambda$).

Let's put these probabilities together, using a 'fate diagram' (following Brownie *et al*. 1985).



Thus, recovery data supplies information directly (and directly is the key operative word here) about only those birds which are shot and reported. Thus, under this parameterization, not everything is estimable - only the product $Kc\lambda$ is estimable, but the component probabilities K, c and λ are not. Generally, the product $Kc\lambda$ (often written as $H\lambda$, where $H = Kc$ = harvest rate; the probability of being killed and retrieved by a hunter during the year) is referred to as the recovery rate, f.* Using these 'product' (summary) parameters, we can modify the preceding 'fate diagram' as follows:



* We note that neither 'harvest rate' or 'recovery rate' are 'rates' in the strict sense of the word (which implies instantaneous rates of change). Strictly speaking, they should be referred to as 'harvest probability' and 'recovery probability', respectively. However, the use of the word 'rate' is traditional for these models.

Different assumptions about the parameters f and S give rise to the different models. In this sense, you can loosely (very loosely) think of f and S as the equivalents of p and φ for a live recapture analysis - clearly not in terms of what they represent, but in the fact that the 'encounter history' is defined by these 2 probabilities. Remember, the components of the recovery probability f (i.e., $Kc\lambda$) are not estimable without additional information (discussed later).

Let's see how these two primary parameters (f and S) combine to determine the expected numbers of bands recovered in a particular time period. The process is analogous to expressing the expected numbers of individuals with capture history '101101' as a function of the number released (R) and the underlying survival and recapture probabilities.

Suppose N_1 individuals are marked. How many recoveries are expected during the next year? Note, we're not asking specifically how many individuals are alive at the end of the 12 months following marking (although this can be derived, obviously), but rather, how many individuals will be (i) shot by hunters, (ii) retrieved, and (iii) reported? Look at the fate diagram on the preceding page. The probability that an individual is harvested, retrieved and reported (i.e., the individual is recovered) is simply f. Thus, the expected number of the N_1 released individuals we expect to be recovered in the first interval after marking is given simply as $N_1 f$. If we assume for the moment that both survival and recovery probabilities are time-specific, then the expected number of recoveries are given as follows:

		year recovered			
year marked	*number marked*	1	2	3	$l = 4$
1	N_1	$N_1 f_1$	$N_1 S_1 f_2$	$N_1 S_1 S_2 f_3$	$N_1 S_1 S_2 S_3 f_4$
2	N_2		$N_2 f_2$	$N_2 S_2 f_3$	$N_2 S_2 S_3 f_4$
3	N_3			$N_3 f_3$	$N_3 S_3 f_4$
$k = 4$	N_4				$N_4 f_4$

Make sure you understand the connection between f, S, and the expected number of recoveries. S_i is the probability of surviving from time $(i-1)$ to time (i), whereas f (recovery rate) is the probability of being shot (i.e., not surviving), and then being retrieved and reported. So, f (recovery rate) combines the mortality event with two other events (retrieval and reporting). For example, for individuals marked in year 1, the number of expected dead recoveries in the second interval after marking is given as $N_1 S_1 f_2$. Why? Well, recall that the recovery parameter f is the probability of the mortality event. In order for the individual to be a dead recovery in the second interval, it has to survive the first interval (with probability S_1), and then be harvested, retrieved and reported (with probability f_2). Note that survival S does not appear on the diagonal.

Now, if you've already worked through the earlier chapters on mark-recapture, in looking at the table of expected number of recoveries (above), you probably recognize right away that there are reduced parameter models which can be fit. The expected recoveries shown in the preceding table reflect the expectations from a time-dependent model $\{S_t f_t\}$. Of course, you could fit model $\{S_t f_.\}$ - time dependence in survival only, or model $\{S_. f_.\}$ - constant survival and recovery probabilities, or a whole host of additional models. For the moment, let's quickly run through how you would fit the following 4 models: $\{S_t f_t\}$, $\{S_. f_t\}$, $\{S_t f_.\}$ and $\{S_. f_.\}$.

Historically, a subset of these models have been referred to by generic model names (for example, model $\{S_t f_t\}$ is referred to in Brownie *et al*. (1985) as Model 1). In the following, we note this historical connection - we suggest that in general you use a explicit model naming convention as we've used throughout the book (and as suggested in Lebreton *et al*. 1992). However, it is important to understand the historical naming conventions to allow you to easily read and interpret earlier papers and texts.

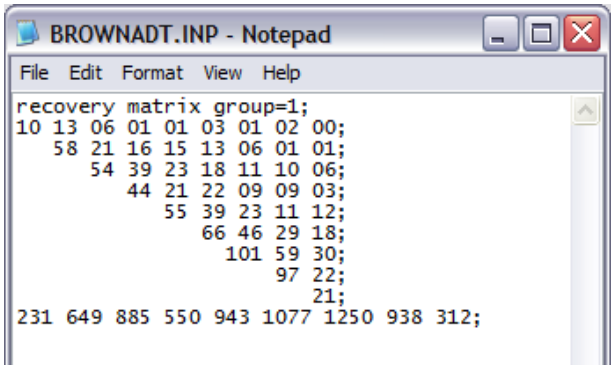
For individuals marked as adults (i.e., no implicit age-classification), our models (and their corresponding legacy names) are:

model	*legacy name*	*reference*
$\{S_t f_t\}$	Model 1	Brownie *et al*. (1985) pp. 15-20
$\{S_t f_.\}$	none	
$\{S_. f_t\}$	Model 2	Brownie *et al*. (1985) pp. 20-24
$\{S_. f_.\}$	Model 3	Brownie *et al*. (1985) pp. 24-30

You might be wondering about model $\{S_t f_.\}$? There is no model in the Brownie *et al*. (1985) methodology corresponding to this model because this model (which assumes the recovery probability f is constant over time, while survival S varies) is seldom applicable to the waterfowl data sets for which the model set in Brownie *et al*. (1985) was developed.

To demonstrate how to fit these models using **MARK**, we'll use data set `BROWNADT.INP` (a subset of the `BROWNIE.INP` data file distributed with **MARK**). `BROWNADT.INP` contains the recovery data for adult male mallards marked in the San Luis Valley in Colorado, from 1963 to 1971. The full data set (`BROWNIE.INP`) contains data for both the adults and juveniles. For the moment, we'll look only at the adults.

Start **MARK**, and begin a new project by pulling down the '**File**' menu and selecting '**New**'. Select the file `BROWNADT.INP`. Before we go any further, let's have a look at the file. Again, the easiest way to do this is to click the '**View file**' button. Here's what `BROWNADT.INP` looks like:

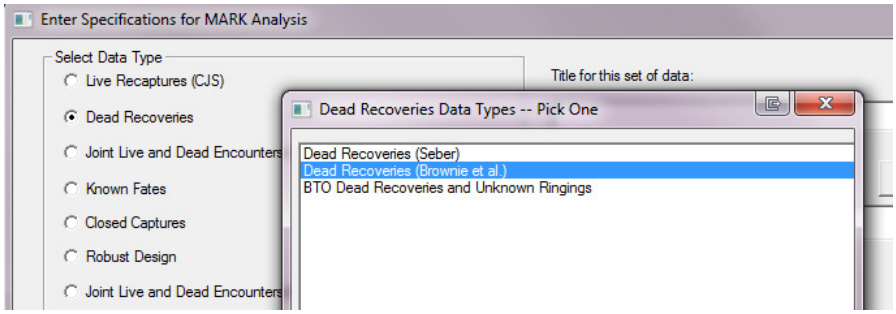


We see that the data are stored in 'classic' recovery matrix form. It is not necessary to format the data this way for a recovery analysis, but it is a traditional summary format. However, remember that using any sort of summary format, whether for a recovery analysis or for (say) mark-recapture analyses has the major disadvantage of not allowing individual covariates (since all individuals are lumped together in the summary). The other approach is to use the familiar encounter history format. **MARK** makes use of what we refer to as the 'LDLD' format to code dead recovery data (and joint live encounter-dead recovery data – this data type is covered in chapter 9). For more details on the LDLD format, see chapter 2.

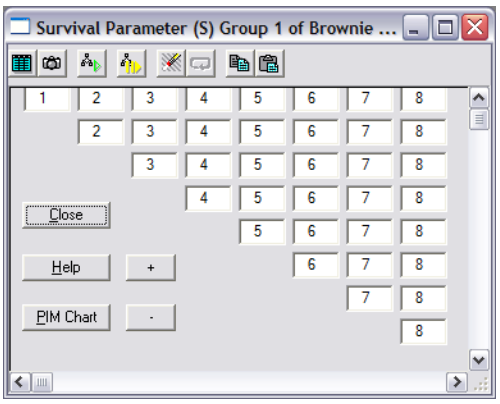
There are 9 'occasions' in this data set, although we submit that occasions is not particularly useful as a reference term, since it is not accurate. In mark-recapture, the occasion is used to refer to the point in time (i.e, the occasion) upon which a marked individual was encountered. Occasions were separated by intervals. In recovery analysis, the data refer to the total number of individuals recovered during the interval, and not at a particular occasion. Thus, it is probably more appropriate to refer to the intervals

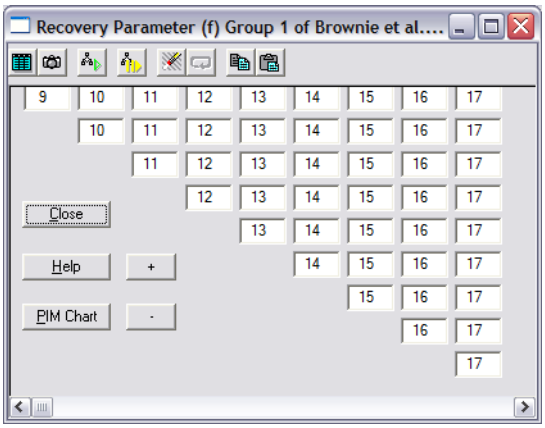
themselves. In this example, we have 9 years ($l = 9$) of recovery data (as it turns out, ranging from 1963 to 1971). The bottom row indicates the number of newly marked individuals released at the start of each year (note that the year doesn't necessarily start with January 1 - it could be that 'year' refers to the 12-month interval between hunting seasons, for example). So, at the start of what we refer to as 1963, 231 newly marked adult mallards were released. Of these, 10 were recovered during the first 12 months following this release, 13 were recovered the next year, and so forth. Birds were marked and released each year of the study - in other words, there are $k = 9$ rows of recovery data in the data file (i.e., the recovery matrix is symmetric, $k = l$). This becomes important later on, so keep the fact that '$k = l$' in the back of your mind. Set the number of encounter occasions in **MARK** to 9.

Now we need to select the data type. Remember, **MARK** 'can't tell' the sort of data (or analysis) you are interested in from the data – you have to 'tell it'. Now, if you look at the data type list in the **MARK** specification window, you'll see a radio-button corresponding to '**`Dead Recoveries`**'. If you select this radio-button, a small window will pop up as you to pick a dead recovery data type. Three are listed: '**`Dead Recoveries (Seber)`**', '**`Dead Recoveries (Brownie et al.)`**', and '**`BTO Dead Recoveries and Unknown Ringings`**'.



We're starting with the '**`Brownie`**' approach, even though it is not the first one presented in the **MARK** data type menu, simply because it is the 'classic' approach used in the vast majority of published recovery analysis. So, as shown, select the '**`Dead Recoveries (Brownie et al.)`**' data type from the list, and then click the '**`OK`**' button. You should now see the survival (S) PIM on the screen (just as with live encounter - recapture - data, **MARK** defaults to opening up the 'survival' PIM). However, there are some subtle but important differences between the survival and recovery PIMs, at least when using the Brownie parameterization. To explore this, let's also open up the recovery (f) PIM for comparison.





Woah – wait a second! These two PIMs don't have the same number of rows and columns - is this a mistake?! No! This is exactly the way it should be. Of course, now you need to consider *why* this is true. Look again at the table of expected recoveries, and the associated probability expressions, we saw earlier. In that example, we were considering only 4 years ($l = k = 4$), but the principle is exactly the same.

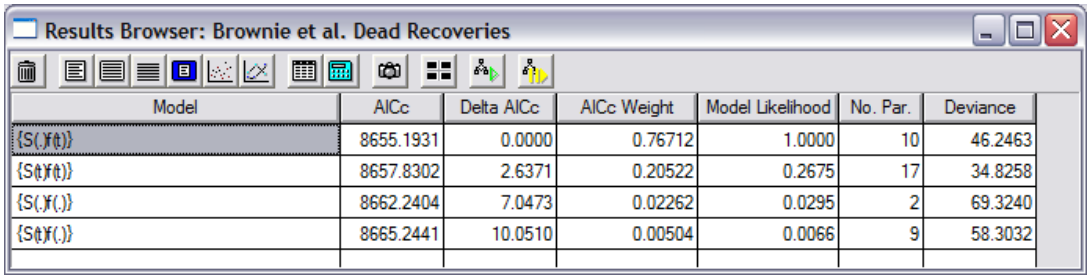
		year recovered			
year marked	*number marked*	1	2	3	$l = 4$
1	N_1	$N_1 f_1$	$N_1 S_1 f_2$	$N_1 S_1 S_2 f_3$	$N_1 S_1 S_2 S_3 f_4$
2	N_2		$N_2 f_2$	$N_2 S_2 f_3$	$N_2 S_2 S_3 f_4$
3	N_3			$N_3 f_3$	$N_3 S_3 f_4$
$k = 4$	N_4				$N_4 f_4$

The key is to look carefully at the probability expressions in each cell. Remember that in the case of live mark-recapture, the PIMs are (in effect) constructed from the subscripts of the parameters in the corresponding probability expressions. What about for dead recovery analysis? Look at the subscripting of the two primary parameters, S and f. If you look along the first row (the row with the greatest number of columns - years), we see that the subscripting for recovery probability f ranges from '1' to '4'. In contrast, we see that the subscripting for survival, S, ranges from '1' to '3' only. Thus, the PIM for S will necessarily be 'smaller' (i.e., reduced dimension) than the PIM for recoveries.

Make sure you understand why - the key is in the first year following the release of newly marked individuals. Consider the first cohort, where N_1 individuals are marked and released. As noted earlier, during that first year after marking and release, the expected number of individuals recovered is $N_1 f_1$ - there is no S term since S denotes survival. An individual cannot survive the interval and also be recovered during the interval (since a recovery implies mortality). The survival term S shows up only in years after the first year following marking (i.e., years 2, 3, 4...). Why? Again, as noted earlier, this is because in order to be recovered in (say) year 2 after marking, the individual must have survived year 1 (thus, the expected number of recoveries in the second year after marking is $N_1 S_1 f_2$).

Now, with a bit of thought, you might think that these 'asymmetric' PIMs might have implications for which parameters are individually identifiable. You would be correct – more on parameter identifiability in a moment. For now, let's proceed and run this model (we'll call it model '`S(t)f(t)`').

If you've worked through the preceding chapter of this book, it should be immediately obvious how to fit the other models in our candidate model set (again, the most efficient way is by manipulating the PIM chart). Go ahead and run the remaining 3 models, and add the results to the browser.



Model	AICc	Delta AICc	AICc Weight	Model Likelihood	No. Par.	Deviance
{S(.)f(t)}	8655.1931	0.0000	0.76712	1.0000	10	46.2463
{S(t)f(t)}	8657.8302	2.6371	0.20522	0.2675	17	34.8258
{S(.)f(.)}	8662.2404	7.0473	0.02262	0.0295	2	69.3240
{S(t)f(.)}	8665.2441	10.0510	0.00504	0.0066	9	58.3032

We see clearly that model $\{S_{.}f_t\}$ (Model 2 *sensu* Brownie *et al*. 1985) and model $\{S_t f_t\}$ (i.e., Model 1 *sensu* Brownie *et al*. 1985) are the 'best' two models out of the four in the model set (since they are clearly better supported by the data than are the other two models). Among these two models, model $\{S_{.}f_t\}$ is almost 4 times better supported by the data than is the fully time-dependent model $\{S_t f_t\}$. Using the classical 'model comparison' paradigm, the LRT between these two models confirms the 'qualitative result' from comparisons of the Akaike weights; the fit of model $\{S_{.}f_t\}$ was not significantly different from that of model $\{S_t f_t\}$ ($\chi^2 = 11.42$, $P = 0.121$), so we accept model $\{S_{.}f_t\}$ as our most parsimonious model, and conclude there is no 'significant' evidence of time-dependence in survival in these data.

Now we come to to the first challenge of the exercise - which we hinted at somewhat in the discussion of the 'asymmetry' of the PIMs (above). How are the number of parameters determined? Which parameters are identifiable in each of the models?

8.2. Counting parameters - Brownie parameterization

Let's start by having yet another look at the table of expected recoveries for the simpler 4 year study.

		year recovered			
year marked	*number marked*	1	2	3	$l = 4$
1	N_1	$N_1 f_1$	$N_1 S_1 f_2$	$N_1 S_1 S_2 f_3$	$N_1 S_1 S_2 S_3 f_4$
2	N_2		$N_2 f_2$	$N_2 S_2 f_3$	$N_2 S_2 S_3 f_4$
3	N_3			$N_3 f_3$	$N_3 S_3 f_4$
$k = 4$	N_4				$N_4 f_4$

As structured, this corresponds to model $\{S_t f_t\}$ - full time-dependence in both parameters. How many of these parameters are identifiable? The key to answering this question is to see whether or not there are any 'groups' of parameters that always occur together, and never apart. In the preceding table, we see that no such 'groups' exist - every parameter ($S_1 \rightarrow S_3$) and ($f_1 \rightarrow f_4$) occurs either alone or in unique combinations. As such, all 7 parameters are identifiable. In general, for model $\{S_t f_t\}$, the number of identifiable parameters is $2k-1$ (where k is the number of release cohorts). However, as we'll see in a minute, this isn't always the case.

What about model $\{S_{.}f_t\}$? The probability statements for this model are:

		year recovered			
year marked	*number marked*	1	2	3	$l = 4$
1	N_1	$N_1 f_1$	$N_1 S f_2$	$N_1 S S f_3$	$N_1 S S S f_4$
2	N_2		$N_2 f_2$	$N_2 S f_3$	$N_2 S S f_4$
3	N_3			$N_3 f_3$	$N_3 S f_4$
$k = 4$	N_4				$N_4 f_4$

In this case, all 5 parameters are identifiable - S and ($f_1 \rightarrow f_4$).

Now, at this point you might be saying 'Gee...in both cases, all the parameters are identifiable...is this always the case?'. If only life were that simple! Consider the situation shown at the top of the next page.

		year recovered			
year marked	*number marked*	1	2	3	$l = 4$
1	N_1	$N_1 f_1$	$N_1 S_1 f_2$	$N_1 S_1 S_2 f_3$	$N_1 S_1 S_2 S_3 f_4$
2	N_2		$N_2 f_2$	$N_2 S_2 f_3$	$N_2 S_2 S_3 f_4$
k=3	N_3			$N_3 f_3$	$N_3 S_3 f_4$

The first notable feature is that $k \neq l$ (i.e., the number of rows in the recovery matrix, $k = 3$, is less than the number of columns - years of the study, $l = 4$). This sort of situation is not that uncommon. Marking individuals can be time consuming, and expensive, but collecting the recovery data is passive, inexpensive (generally), and continues as long as there is hunting - often long after the marking is completed. In this case, recovery data were collected for $s = 2$ years ($s = l - k$) after the cessation of marking.

begin sidebar

Formatting the recovery matrix when $k \neq l$

When $k \neq l$ (typically when the number of years of marking is less than the number of years over which recovery data are collected - i.e., $k < l$), does this influence the structure if the data `.INP` file? The answer, as you may recall from Chapter 2, is 'yes'. You need to add '0's for the 'missing elements' of the recovery matrix. For example, if $k = 3$, $l = 5$, the recovery matrix would look like:

$$\begin{array}{ccccc} R_1 & R_2 & R_3 & R_4 & R_5; \\ & R_2 & R_3 & R_4 & R_5; \\ & & R_3 & R_4 & R_5; \\ & & & 0 & 0; \\ & & & & 0; \\ N_1 & N_2 & N_3 & 0 & 0 \end{array}$$

end sidebar

Now, if you read Brownie *et al*. (1985), you'd eventually come to a point where you're told

> *'In general, under Model 1 (i.e., $S_t f_t$), the parameters $f_1, f_2, \ldots, f_k$ and $S_1, S_2, \ldots, S_{k-1}$ are separately estimable, but if $s > 0$ (where $s = l - k$), only products such as $S_k f_{k+1}$, $S_k S_{k+1} f_{k+2}, \ldots, S_k S_{k+1}, \ldots, S_{k+s-1} f_{k+s}$ are also estimable, not the individual parameters S_{k+j-1}, and f_{k+j}, $j = 1, \ldots, s$.'*

OK, now to translate – look carefully at the table of probability expressions at the top of this page (for the time-dependent model $\{S_t f_t\}$, where $k < l$). We mentioned previously that the key to identifying inestimable parameters is to look for 'groups' of parameters that are never separated. Do we have any in this table? In fact, we do in this case. Notice that the parameters S_3 and f_4 always occur together as the product $S_3 f_4$ (i.e., whenever you find f_4 you always find S_3). So, they are not separately identifiable.

But you might say 'Well, S_2 and f_3 always occur together, as do S_1 and f_2, so are they identifiable?'. The answer in those cases is 'yes', because for those years (3 and 2, respectively), the last element of the column is simply the product of the number released and the recovery probability – no survival term. In contrast, in column 4, every element of the column has the products of the survival and recovery probabilities. Why does this matter? It matters because it is these final elements of the columns 2 and 3 which allow you to estimate the various parameters. Also, with $k = 3$, columns 1 to 3 correspond to

$l = 3$ (i.e., form a symmetrical recovery matrix), and thus all parameters are identifiable. In column 4, this is not the case, since all elements of column 4 contain at least one product in common ($S_3 f_4$).

Thus, in this example, S_1 and f_2 are separately identifiable, as are S_2 and f_3, but only the products $S_3 f_4$ is identifiable, so 5 parameters in total (4 individual, and 1 product). In general, estimates of the products are not of particular interest, since, for example, $S_3 f_4$ is the probability of surviving year 3 and being shot and reported in year 4.

However, non-identifiability can 'vanish' with a reduction in complexity of the model. You may recall this from the mark-recapture chapters, where non-identifiability did not occur in reductions from the fully time-dependent model. The same is true here. If survival probability S is constant over time, for example, then

		year recovered			
year marked	*number marked*	1	2	3	$l = 4$
1	N_1	$N_1 f_1$	$N_1 S f_2$	$N_1 S S f_3$	$N_1 S S S f_4$
2	N_2		$N_2 f_2$	$N_2 S f_3$	$N_2 S S f_4$
k=3	N_3			$N_3 f_3$	$N_3 S f_4$

In this case, because estimation of S is based on data from all years, there is no problem on non-identifiability - both S and all of the recovery parameters are estimable.

However, although everything is estimable, Brownie *et al*. (1985) notes that for years $> k$, estimates of recovery probability tend to be poor, because they are based on so few data. So, in this example, f_4 would be poorly estimated, since they are based entirely on recoveries from > 1 year after marking. At this point, we'll introduce some nomenclature in common use in the literature. Recoveries that occur during the year following marking are referred to as *direct recoveries*, while those that occur > 1 year after marking are referred to as *indirect recoveries*. More on these 2 'types' of recoveries later.

begin sidebar

Counting parameters in Brownie models: another approach

If you're still confused about how to determine which parameters are estimable in Brownie models, here is another way of approaching the problem which might be more intuitive. Consider the following example recovery matrix, which is based on 4 release occasions:

		year recovered			
year marked	*number marked*	1	2	3	$l = 4$
1	N_1	$N_1 f_1$	$N_1 S_1 f_2$	$N_1 S_1 S_2 f_3$	$N_1 S_1 S_2 S_3 f_4$
2	N_2		$N_2 f_2$	$N_2 S_2 f_3$	$N_2 S_2 S_3 f_4$
3	N_3			$N_3 f_3$	$N_3 S_3 f_4$
k=4	N_4				$N_4 f_4$

We'll introduce the approach by considering two 'problem' situations - (1) no recoveries in a given year, and (2) no mark-release effort in a given year.

We'll consider the problem of no recoveries in a given year first. For the preceding recovery matrix, the most direct way to get an estimate of S_1 is *algebraically*, by comparing the two cells in column 2 of the recovery matrix (above). You have information on f_2 from direct recoveries (along the diagonal), and information on the product of $S_1 f_2$ (based on the indirect recoveries from the first release cohort). This constitutes two equations in two unknowns, which is easily solved for S_1. If $f_2 = 0$ (as would be the case if there were no recoveries in year 2 of the study), then there is no information on S_1 in column 2. However, looking at column 3, you can derive an estimate of S_1 from the combination of

data from the top two cells in this column, and derive an estimate of S_2 from the combination of data from the bottom two cells in that column, assuming that there were recoveries in year 3. Normally you would not do these things, because S_1 and S_2 are also found in other cells in the model. The Brownie models use all information from all of the cells in the recovery matrix, to maximize precision. However, this 'algebraic' approach at least tells you whether the minimum data necessary for estimation of a particular parameter are available, given the absence of recoveries in one or more years of the study.

A somewhat more difficult problem arises when you also have years where you do not release any animals. In this case you are taking out an entire row of the recovery matrix - for example, as shown in the following recovery matrix:

		year recovered			
year marked	*number marked*	1	2	3	$l = 4$
1	N_1	$N_1 f_1$	$N_1 S_1 f_2$	$N_1 S_1 S_2 f_3$	$N_1 S_1 S_2 S_3 f_4$
2	N_2		0	0	0
3	N_3			$N_3 f_3$	$N_3 S_3 f_4$
k=4	N_4				$N_4 f_4$

In this example, we did not release any animals in year 2, and thus an entire row of the recovery matrix is set to 0. In this case, if you use the same algebraic approach described above, you will see that you lose your ability to estimate S_1 and S_2. You don't have direct recovery information on f_2 and therefore cannot use it to extract S_1 from the product $S_1 f_2$. In addition, you lose information on the product $S_2 f_3$, and therefore again cannot use it to algebraically 'solve' for S_2. The best you can do in this case is estimate the product $S_1 S_2$. In general, then, when you do not release animals in year t, you cannot get separate estimates of S_{t-1} and S_t.

end sidebar

Now that we've had a brief look at some of the considerations for counting parameters under the Brownie parameterization, let's return to the adult mallard example we have been working with. At this point, you should be able to figure out why model $\{S_t f_t\}$ (for example) has 17 identifiable parameters. Since $k - l = 9$, then we have $k + l - 1$ identifiable parameters: 9 recovery probabilities, and 8 survival rates. For model $\{S_\cdot f_t\}$ we have 10 identifiable parameters: 1 survival rate, and 9 recovery probabilities.

8.3. Brownie estimation: individuals marked as young only

In the preceding mallard example, we noted in passing that the data set consisted entirely of individuals marked as adults. What happens if you face the situation where you have only individuals marked as young? Can you still estimate survival and recovery probabilities? Are all parameters identifiable?

This general question is dealt with thoroughly in Brownie *et al.* (1985), pp. 112-115, and the associated paper by Anderson, Burnham & White (1985), reprinted in full as an Appendix in Brownie *et al.* (1985). These references should be consulted for a full treatment of the problem. Our motive here then is to 'test you' on your ability to determine which parameters are identifiable, and which are not. Paraphrasing Brownie *et al*. (1985), marking of young individuals only is often popular because it is often easier, and less expensive (young are typically easier to catch than adults or sub-adults). However, in most cases (perhaps even in all cases), survival of young individuals is typically lower than the survival of older age classes, Also, first year (direct) recovery probabilities are typically higher than for older, adult individuals (this is to some degree a logically consistent statement, since some mortality, the complement of survival, is 'hidden' in recovery rate).

Given this, we first need to consider what an appropriate model would be for modeling recoveries

from a sample of individuals marked as young. Brownie *et al*. (1985) describe a '`model H1`' as an appropriate model for these sorts of data (pp. 59-62). Its structure is shown below for a situation where $k = l = 4$.

		year recovered			
year marked	*number marked*	1	2	3	$l = 4$
1	N_1	$N_1 f_1^*$	$N_1 S_1^* f_2$	$N_1 S_1^* S_2 f_3$	$N_1 S_1^* S_2 S_3 f_4$
2	N_2		$N_2 f_2^*$	$N_2 S_2^* f_3$	$N_2 S_2^* S_3 f_4$
3	N_3			$N_3 f_3^*$	$N_3 S_3^* f_4$
$k = 4$	N_4				$N_4 f_4^*$

Basically, `model H1` is model $\{S_{a2-t/t} f_{a2-t/t}\}$ - an age structured model with 2 age-classes with time-dependence for each class. If you worked through the preceding chapters on mark-recapture (Chapter 7 in particular), you should quickly recognize this structure, at least qualitatively. Along the diagonal, the recovery probabilities (denoted with an asterisk, *) reflect the recovery probabilities for young individuals, whereas the off-diagonal recovery probabilities (no asterisk) refer to recovery probabilities for adult age classes (remember that time, and thus age, increase going from left to right within a cohort - along a row). The survival rates marked with an asterisk (which form an internal diagonal within each column) represent survival during the first year of young individuals.

Now, what (if anything) can be estimated here? With a bit of thought, and looking carefully at the preceding table, you should see that the direct recovery probabilities f^* are estimable (recall that direct recovery probabilities are the recovery probabilities estimated for the first interval following marking). However, without extra information, S^* - the survival probabilities of young over the interval following marking - are not estimable, no matter what simplifying assumptions are made about how the probabilities vary over time.

Remember the trick is to look for parameter 'groups' that always occur together. Consider the following attempt to simplify the structure of this model in an attempt to 'make the parameters identifiable'. Assume that none of the 4 parameters (S, S^*, f and f^*) vary over time (i.e., model $S_{a2-./.} f_{a2-./.}$). The structure of this model would be (again assuming $k = l = 4$):

		year recovered			
year marked	*number marked*	1	2	3	$l = 4$
1	N_1	$N_1 f^*$	$N_1 S^* f$	$N_1 S^* S f$	$N_1 S^* S S f$
2	N_2		$N_2 f^*$	$N_2 S^* f$	$N_2 S^* S f$
3	N_3			$N_3 f^*$	$N_3 S^* f$
$k = 4$	N_4				$N_4 f^*$

Note that the parameters S^* and f always occur together as a product. In fact, this demonstrates why, even in this simple model, these two parameters cannot be separately estimated – only the product $S^* f$ can ever be estimated if no adults are banded. *Moral*: don't mark only young individuals if you plan on using a recovery analysis alone to estimate parameters of interest - it is doomed to fail. (An approach combining data from dead recoveries and live encounters applied to individuals marked as young only is described in the next chapter).

8.4. Brownie analysis: individuals marked both as young + adults

One of the unintended (yet important) messages of the preceding section was that recovery analysis of only individuals marked as young is ultimately futile. Of course, you should also understand that this statement is true only for recovery analysis - at least when contrasted to standard mark-recapture analysis, which has no such structural limits.

But, the question remains - how can you get age-specific estimates from a recovery analysis? The answer is, in fact, fairly straightforward - you mark both young and adults, and analyze their recovery data together. The reason we do this (as we'll see in a moment) is that the 'extra information' provided from the adults allows us to estimate some parameters we wouldn't be able to estimate using young alone.

The background for analyzing individuals marked both as young and adults using the Brownie parameterization is found in Brownie *et al*. (1985) - see pp. 56-115. As in Brownie *et al*. (1985), we'll start with a very general model - what is referred to as '`Model H1`' in the Brownie text (which we introduced in the preceding section). `Model H1` assumes (1) that annual survival, reporting and harvest probabilities are year-specific, (2) annual survival and harvest probabilities are age-dependent for the first year of life only, and (3) reporting probabilities are not dependent on the time of release.

As with the preceding discussion on individuals marked as young only, we'll let f_i^* be the recovery probability in year (i) for individuals marked and released as young in year (i). S_i^* will represent the survival rate for year (i) for individuals marked and released as young in year (i). f_i and S_i will represent the adult recovery and survival rates in year (i), respectively. Now, let's examine the structure of `Model H1`, again using a table of the probability expressions corresponding to the number of expected direct and indirect band recoveries.

		year recovered			
Year marked	*Number marked*	1	2	3	$l = 4$
		marked and released as adults			
1	N_1	$N_1 f_1$	$N_1 S_1 f_2$	$N_1 S_1 S_2 f_3$	$N_1 S_1 S_2 S_3 f_4$
2	N_2		$N_2 f_2$	$N_2 S_2 f_3$	$N_2 S_2 S_3 f_4$
3	N_3			$N_3 f_3$	$N_3 S_3 f_4$
k=4	N_4				$N_4 f_4$
		marked and released as young			
1	M_1	$M_1 f_1^*$	$M_1 S_1^* f_2$	$M_1 S_1^* S_2 f_3$	$M_1 S_1^* S_2 S_3 f_4$
2	M_2		$M_2 f_2^*$	$M_2 S_2^* f_3$	$M_2 S_2^* S_3 f_4$
3	M_3			$M_3 f_3^*$	$M_3 S_3^* f_4$
k=4	M_4				$M_4 f_4^*$

For marked adults, the assumptions of `Model H1` are the same as those of `Model 1` (i.e., model $S_t f_t$), so the expected recoveries from individuals marked as adults are the same under `Model H1` and `Model 1`.

For individuals marked as young, if M_1 are marked and released in the first year, then on the average we would expect $M_1 f_1^*$ recoveries in the first year after marking, and $M_1 S_1^*$ of the release cohort to survive to adulthood (i.e., to survive the year). At the start of the second year, M_2 new individual young are marked and released. In addition, the $M_1 S_1^*$ survivors from the first release cohort (now adults) are also released. The important thing to remember is that in the second year, these $M_1 S_1^*$ survivors will

reflect the adult probabilities f_2 and S_2, giving on average $M_1 S_1^* f_2$ recoveries and $M_1 S_1^* S_2$ survivors. And so on for each successive cohort and recovery year.

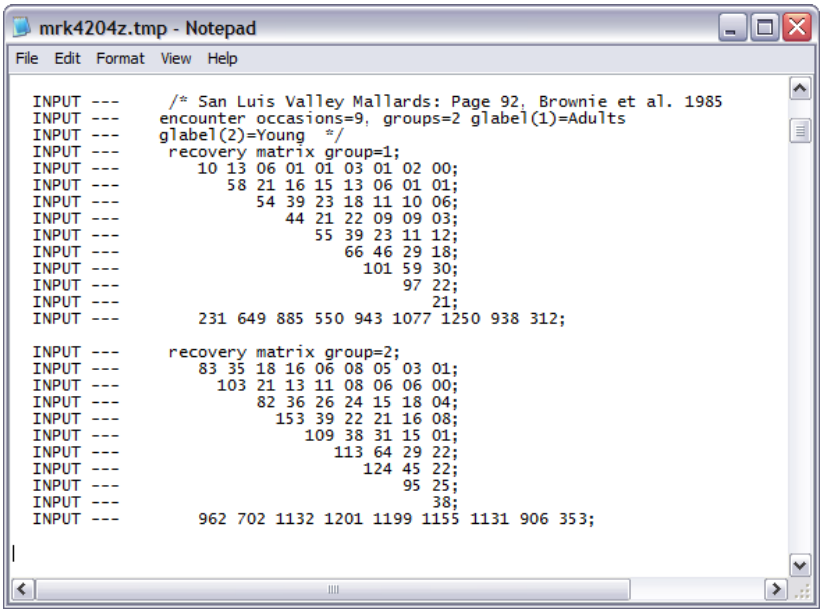
From the table of expected recoveries for `Model H1` we see that the off-diagonal elements of the recovery matrix for individuals marked as young provide information about the adult probability parameters.

		year recovered			
Year marked	*Number marked*	1	2	3	$l = 4$
			marked and released as young		
1	M_1	$M_1 f_1^*$	$M_1 S_1^* f_2$	$M_1 S_1^* S_2 f_3$	$M_1 S_1^* S_2 S_3 f_4$
2	M_2		$M_2 f_2^*$	$M_2 S_2^* f_3$	$M_2 S_2^* S_3 f_4$
3	M_3			$M_3 f_3^*$	$M_3 S_3^* f_4$
k=4	M_4				$M_4 f_4^*$

It is the presence of the 'adult' parameters in the off-diagonal cells that can be exploited to provide extra information needed to estimate parameters that might not be estimable otherwise.

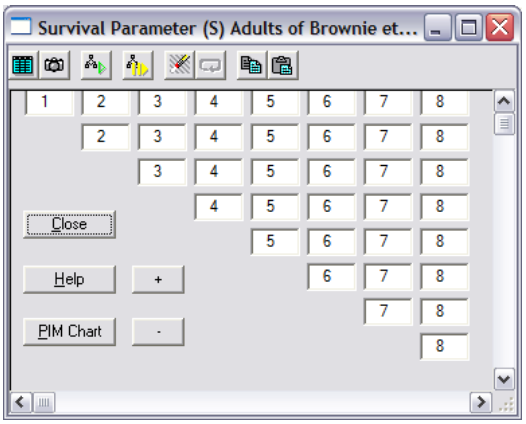
Let's now turn our attention to running `Model H1` in **MARK**. In fact, when you installed **MARK**, you'll find that this model (and 2 others) have already been 'done for you'. During the installation, a file called `BROWNIE.DBF` was extracted into the directory where **MARK** was installed. Go ahead and find it, open it up. The results in the browser were derived by fitting the 3 models listed to the data in `BROWNIE.INP`, which are in fact the mallard data from San Luis Valley, California we considered before - only now we're looking at both the recoveries for individuals marked as young and adults.

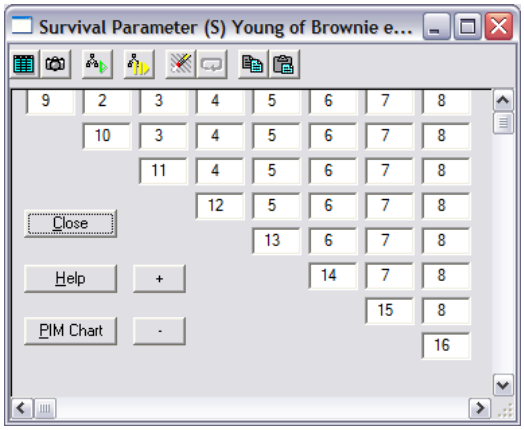
Before we dive into the model structures, and how they're implemented in **MARK**, we should stop for a minute and have a look at the `INP` file format for these data - how do we put both the adult and young recovery matrix into the same input file? As it turns out, it is very simple.



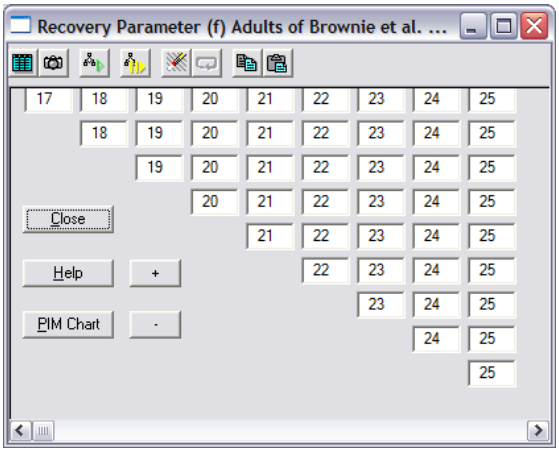
Near the top of the file, you'll see the two recovery matrices, for individuals marked as adults and young, respectively (the order is arbitrary, as long as you remember which one comes first). Note that the two recovery matrices are simply entered sequentially, each one preceded by a '`RECOVERY MATRIX GROUP=n`' statement. That's really all that's needed. The text that is `/* commented */` out is a holdover from the days when these data were run through **BROWNIE** (one of the original programs for running these sorts of data). **MARK** simply ignores the commented out text (as it should).

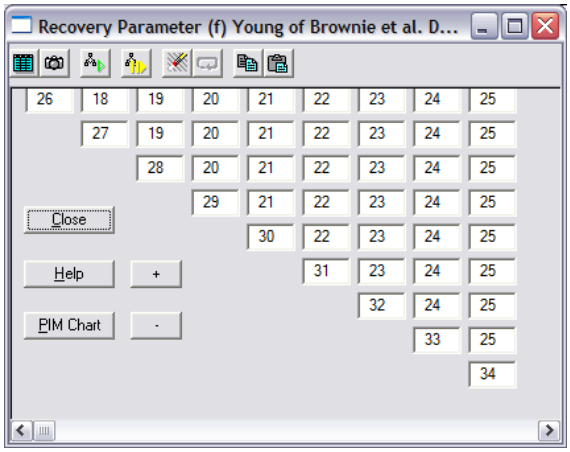
Now let's look at the models themselves. You might guess from inspection of the expected recoveries under `model H1` that this model is in fact model $\{S_{g*(ta2-t/t)-t/t} f_{g*(ta2-t/t)}\}$ - two age classes for both parameters, with time-dependence in each age class. This is model '`S(a*t)f(a*t)`' in the browser (although we prefer a more informative subscripting). The PIMs are shown below starting with survival, S, for adults and young respectively:





Now, the recovery PIMs, again for adults and young, respectively.





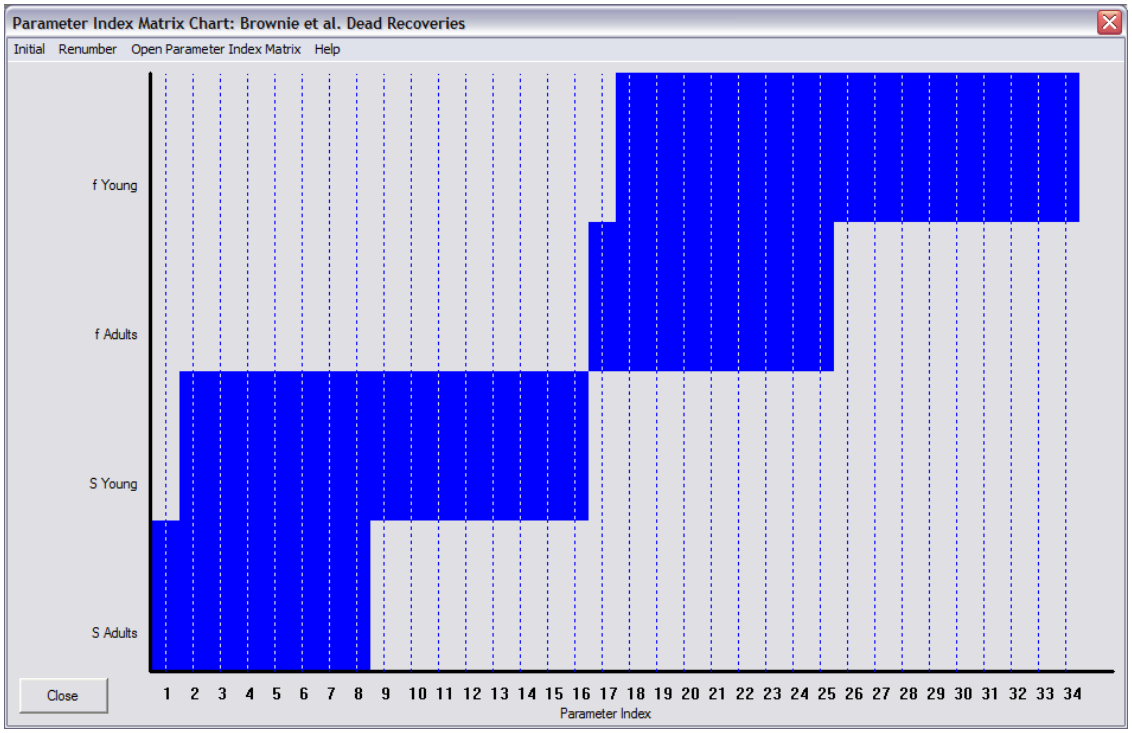
Note that there is no 'age structure' to the adult survival or recovery PIMs. This is because we do not expect differences in the direct recovery or survival rate from the indirect probabilities for

individuals marked as adults. In contrast, note the age-structure for survival and recovery PIMs for individuals marked as young. Again, the age-structure here is because we believe, *a priori*, that survival (and recovery) in the year following marking (i.e., the direct rates), will differ from the probabilities > 1 year after marking (when the surviving individuals are adults).

However, what is important to note here is that the parameter values appear to overlap. Consider the survival PIMs. For individuals marked and released as adults, it is a simple time-dependent PIM, with parameter indexing from 1 → 8. For individuals marked and released as young, there are 2 age-classes. The indexing for the first age-class (along the diagonal) goes from 9 → 16. However, off the diagonal, the indexing ranges from 2 → 8. In other words, off the diagonal, the indexing for the young individuals is the same as that for the adults. Why? Because off the diagonal, individuals marked as young are adults! Remember, time (=age) within cohort goes left to right. You have actually seen this before - it was discussed in some detail in Chapter 7.

Now, implicit in how the PIMs are indexed is the assumption (in this case) that adult survival S_i does **not** depend on whether or not the individual adult released (or entering) at occasion (i) was originally marked as an adult or not. As we will discuss in the next chapter, this may be a 'strong' (i.e., debatable) assumption in some cases.

What about the recovery PIMs? Again, much the same thing - simple time-dependence for adults (indexing ranging from 17 → 25), and age-structure with time-dependence in both age classes for young (26 → 34 along the diagonal for direct recovery probabilities, and 18 → 25 for the adult age-class). To get a different (and perhaps more intuitive view) of the overlapping structure of the PIMs, simply take a look at the PIM chart (shown below), where you can see clearly the overlap between the adult and young PIMs.



Now, let's have a look at the results. Close the PIM chart and click on the '**`View estimates`**' button on the results browser toolbar. Note from the browser that 34 parameters were estimated for this

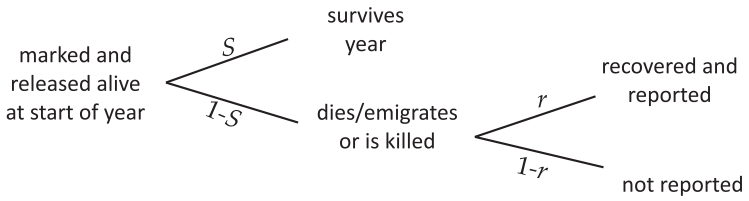
model (`Model H`$_1$). If you look at the PIMs, you'll see that this is the total number of parameters in the structure of the model. Thus, under `Model H`$_1$, when $k = l$ (as it does in this example), all the parameters are estimable - including the young survival and recovery probabilities. Clearly, this is a significant improvement over the case using only individuals marked as young, where essentially nothing was estimable!

8.5. A different parameterization: Seber (S and r) models

In this section we turn our attention to a rather different approach to the same questions, and the same data type - recoveries, but using a different parameterization, first described by Seber (1970) and later by Anderson *et al*. (1985) and Catchpole *et al*. (1995).

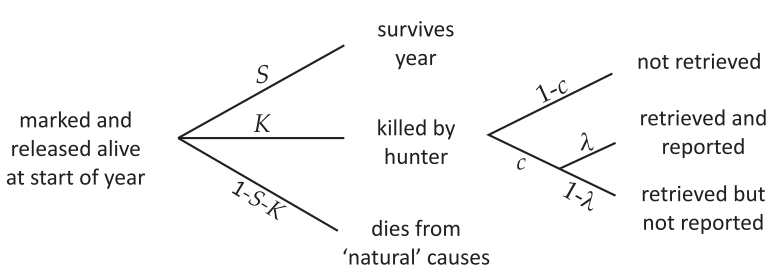
Recall that in the Brownie parameterization we've just covered, marked individuals are assumed to survive from one release to the next with survival probability S_i. Individuals may die during the interval, either due to hunting or due to 'natural' mortality. Individuals dying due to hunting (with probability K_i) may be retrieved and reported with some probability (c_i and λ_i, respectively).

Here, however, we introduce a new parameter r_i, for recovery probability, defined as the probability that dead marked individuals are reported during each period between releases, and (most generally) where the death is not necessarily related to harvest. Note that the recovery parameter r_i we're talking about here is **not** the same as the Brownie recovery probability f_i, which is the probability of being harvested, retrieved and reported during the period between releases.



Thus, a marked individual either (i) survives (with probability S - encounter history '`10`'), (ii) dies and is recovered and reported (with probability $r(1-S)$ - encounter history '`11`'), or (iii) dies and is not reported (either because it was not retrieved, or if retrieved, not reported), with probability in either case of $(1-S)(1-r)$ - encounter history '`10`').

Before we look at how to implement this parameterization in **MARK**, let's take a moment to compare this parameterization with the Brownie parameterization we looked at earlier. First, clearly there must be some logical relationship between r and f. Recall that in the Brownie parameterization,



Consider the encounter history '11'. In the Brownie parameterization, the expected probability of this event is $Kc\lambda$, which we refer to collectively as f, the recovery rate. In the present, modified parameterization, the probability of the encounter history '11' is given as $r(1-S)$. Thus,

$$f_i = r_i(1-S_i) \qquad r_i = \frac{f_i}{(1-S_i)}$$

So, this is the 'formal', algebraic connection between the two parameterizations. Clearly, the parameter r_i is a reduced parameter - and is a function of two other parameters normally found in the Brownie parameterization. But, more pragmatically, what is the impact of the two parameterizations? Why use one over the other, or does it matter?

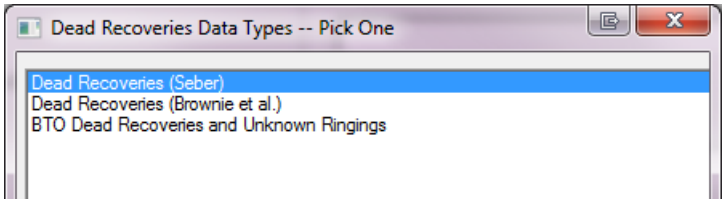
The primary motive for the reduced parameterization (using only S_i and r_i) is so that the encounter process can be separated from the survival process, entirely analogous to what was seen in 'normal' mark-recapture. With the Brownie parameterization, the 2 processes are part of the f_i parameter (i.e., there is 'some survival' and 'some reporting/encounter' information included in recovery rate). As such, developing certain advanced models with **MARK** (by modifying the design matrix) is difficult, even illogical (on occasion) using the Brownie parameterization.

So, we should drop Brownie and use the Seber parameterization right? Well, perhaps not quite. First, under the reduced parameterization the last S_i and r_i are confounded in the time-dependent model, as only the product $(1-S_i)r_i$ (analogous to the confounding of the final $\varphi_i p_{i+1}$ term in fully time-dependent model for live encounter analysis). This has some implications for comparing and contrasting survival estimates for some constrained models - we'll deal with this in a moment.

Second, all the parameters are bounded $[0, 1]$, which outwardly might seem like a benefit. However, parameter estimates at the boundary do not have proper estimates of the standard errors. The Brownie parameterization overcomes both these technical difficulties (for details, see the Brownie text).

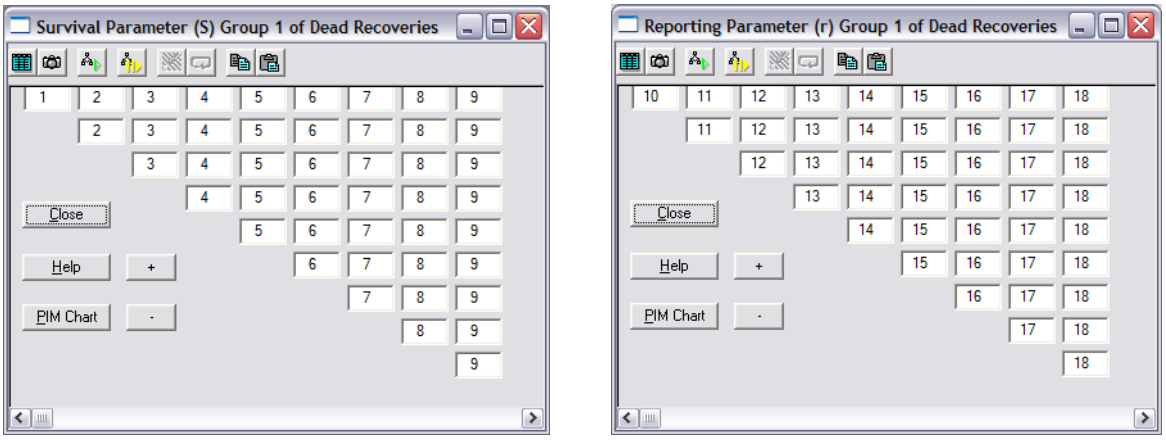
But finally, and perhaps more importantly (at least for some applications), the reduced parameterization does not allow you to separate 'hunter' or 'harvest' mortality from 'natural' mortality, whereas the Brownie parameterization does. The Seber parameterization basically deals with 'mortality' as a whole, with no partitioning possible. In many cases, this can be an important limitation that you need to be aware of.

For the moment, though, we'll leave the comparison of these two models (and their respective pros and cons) for you to explore, and will concentrate on showing how to implement the reduced parameterization in **MARK**. In fact, if you've understood the way in which we applied the Brownie parameterization in **MARK**, you'll find this new approach very easy. We'll demonstrate this using the `BROWNADT.INP` data set we analyzed earlier in the chapter. To specify the new parameterization, select '**`Dead recoveries (Seber)`**':



Look at the PIMs for the two parameters under the fully time-dependent model (shown at the top of the next page). Note that unlike the Brownie parameterization, there are the same number (in absolute terms) of parameters (9) for each ($S_1 \rightarrow S_9$ and $r_1 \rightarrow r_9$).

Since the parameterization is analogous to 'normal' mark-recapture, then the identifiability of parameters should pose no significant challenges for you at this stage. For example, for the fully time-dependent model $\{S_t r_t\}$ with 9 occasions, we expect 17 estimable parameters - $S_1 \rightarrow S_8$ and $r_1 \rightarrow r_8$, and the final product $r_9(1 - S_9)$. If you run this model in **MARK**, you'll see that in fact, 17 parameters are modeled.



8.5.1. Seber vs. Brownie estimates in constrained models: careful!

In the preceding section, we noted that under the Seber parameterization, the last S_i and r_i are confounded in the time-dependent model (analogous to the confounding of the final $\varphi_i p_{i+1}$ term in fully time-dependent model for live encounter analysis). Recall that in live encounter models, the estimate of survival over the final interval can be obtained if the encounter probability on the last occasion is known. We saw that in models where survival varied over time, but encounter probability was held constant (i.e., $\{\varphi_t p_.\}$), all of the survival values were estimable, since a common, constant value for p was estimated for all occasions, including the terminal occasion.

However, while it would seem reasonable to use the same logic for recovery analysis using the Seber parameterization, care must be exercised - especially if you're comparing estimates from a model based on the Seber parameterization with those from a Brownie parameterization. Why? Simple - because the number of survival parameters estimable using the Brownie parameterization is always one less than the Seber parameterization! As such, comparing estimates from a model parametrized using the Seber parameterization can, for some models, be quite different than those from seemingly equivalent models parametrized using the Brownie parameterization.

This can be easily demonstrated by means of a numerical example. Consider the analysis of a simulated data set, 8 occasions, where $K = 0.2$, $c = 1.0$, and $\lambda = 0.4$. In other words, the probability of being harvested ('killed') over a given interval is 0.2, probability of the harvested individual being retrieved is 1.0, and the probability that the harvested, retrieved individual is reported is 0.4. We'll assume all 3 parameters are constant over time. Under the Brownie parameterization, then, the recovery probability is $f = Kc\lambda = (0.2)(1.0)(0.4) = 0.08$.

Given these values, what is the recovery probability r under the Seber parameterization? Recall that

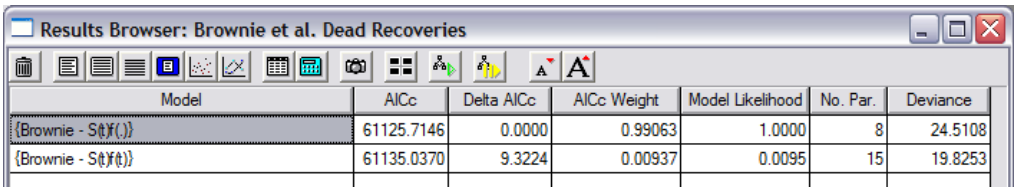
$$f_i = r_i(1 - S_i) \qquad r_i = \frac{f_i}{(1 - S_i)}$$

So, given f from the Brownie parameterization, then we can solve for r provided we have an estimate of S. Since $K = 0.2$, we know that survival probability is at least $(1 - 0.2) = 0.8$. However, this value is derived assuming the only source of mortality is harvest. What if there is some level of natural mortality, say $E = 0.1$? If we assume that harvest and natural mortality events are independent (i.e., temporally separated, or additive), then $S = (1 - K)(1 - E) = (0.8)(0.9) = 0.72$. So, given $f = 0.08$, and $S = 0.72$, then

$$r_i = \frac{f_i}{(1 - S_i)} = \frac{0.08}{(1 - 0.72)} = 0.286$$

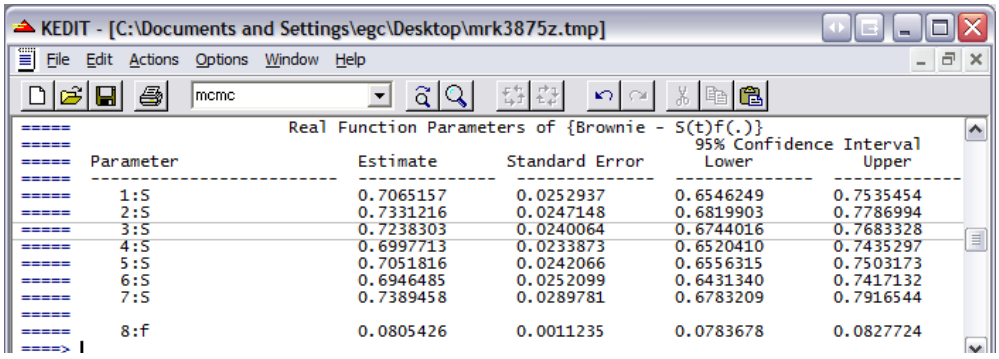
We'll assume no age structure, and 5000 newly marked individuals on each occasion - the recovery data (in `LD` format) are contained in the file `seber-brownie.inp`. We'll start our analysis by specifying the Brownie data type in the data type specification window. If we examine the default starting PIMs for the two parameters, we see that the survival PIM has 7 columns (corresponding to parameters $S_1 \rightarrow S_7$), while the recovery PIM has 8 columns (corresponding to parameters $f_1 \rightarrow f_8$). We run this model (i.e., model $\{S_t f_t\}$), and add the results to the browser.

Then, by modifying the PIMs, we construct a 'constrained' model, $\{S_t f_.\}$, where survival is allowed to vary over time, while the recovery probability is constant (remember - recovery probability under the Brownie parameterization includes information about mortality, since it is the product of kill probability K with the retrieval and reporting parameters c and λ, respectively. As such, a model where recovery probability is held constant, but where survival is allowed to vary over time has likely implications for how 'other sources of mortality' must vary). Model $\{S_t f_.\}$ then has only one recovery estimate, but the same 7 estimates for survival. So, constraining recovery f to be constant over time does not change the number of survival parameters S which are estimable. Here are the results for both models:


Results Browser: Brownie et al. Dead Recoveries

Model	AICc	Delta AICc	AICc Weight	Model Likelihood	No. Par.	Deviance
{Brownie - S(t)f(.)}	61125.7146	0.0000	0.99063	1.0000	8	24.5108
{Brownie - S(t)f(t)}	61135.0370	9.3224	0.00937	0.0095	15	19.8253

We see that model $\{S_t f_t\}$ has 15 estimable parameters (8 recovery parameters + 7 survival parameters), whereas model $\{S_t f_.\}$ has only 8 estimable parameters (1 recovery parameter + 7 survival parameters). If we look at the parameter estimates from model $\{S_t f_.\}$


KEDIT - [C:\Documents and Settings\egc\Desktop\mrk3875z.tmp]

File Edit Actions Options Window Help

mcmc

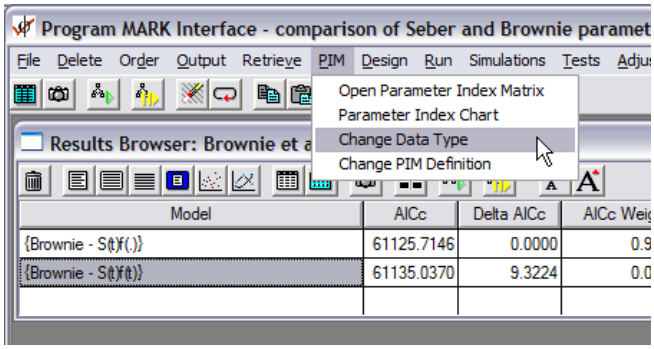
Real Function Parameters of {Brownie - S(t)f(.)}

Parameter	Estimate	Standard Error	95% Confidence Interval Lower	Upper
1:S	0.7065157	0.0252937	0.6546249	0.7535454
2:S	0.7331216	0.0247148	0.6819903	0.7786994
3:S	0.7238303	0.0240064	0.6744016	0.7683328
4:S	0.6997713	0.0233873	0.6520410	0.7435297
5:S	0.7051816	0.0242066	0.6556315	0.7503173
6:S	0.6946485	0.0252099	0.6431340	0.7417132
7:S	0.7389458	0.0289781	0.6783209	0.7916544
8:f	0.0805426	0.0011235	0.0783678	0.0827724

we see that the survival estimates are all fairly close to the true value of 0.72 (recall that in the true model under which the data were simulated, the true value for survival did not vary over time), and

the estimated recovery probability is very close to the true value of 0.08. This is perhaps not surprising given the size of the data set, and that our fitted model is fairly close to the true model underlying these simulated data.

Ok - fine. But now let's fit these same data using the Seber parameterization. We can do this easily in **MARK** by changing the data type from Brownie to Seber. We do this by selecting '**`Change data type`**' from the PIM menu:



and selecting the '**`Dead recoveries`**' data type from the list.

Now, if we examine the PIMs for the fully time-dependent model (i.e., model $\{S_t r_t\}$), we see that the PIMs for both parameters have 8 columns, corresponding to 8 parameters for survival, and 8 parameters for reporting rate, respectively. However, we also recall that the final two estimates of survival and reporting probabilities are confounded under the Seber parameterization, so we in fact have only 15 estimable parameters in this model. Fit this model to the data, and add the results to the browser.

Results Browser: Dead Recoveries

Model	AICc	Delta AICc	AICc Weight	Model Likelihood	No. Par.	Deviance
{Brownie - S(t)f(.)}	61125.7146	0.0000	0.98144	1.0000	8	24.5108
{Brownie - S(t)f(t)}	61135.0370	9.3224	0.00928	0.0095	15	19.8253
{Seber - S(t)r(t)}	61135.0370	9.3224	0.00928	0.0095	15	19.8253

Notice that models $\{S_t f_t\}$ (Brownie) and $\{S_t r_t\}$ (Seber) have **exactly** the same model deviances (19.8253), and number of estimated parameters (15). And, not surprisingly perhaps given this, you'll see that the estimates of survival from the Seber model are *identical* to those from the Brownie model, for the first seven estimates; the final estimate of survival from the Seber model is confounded with the final estimate of the reporting rate.

OK - so far, it seems as if the two parameterizations are equivalent. Ah, not so fast. Now, let's try model $\{S_t r_.\}$ using the Seber parameterization. Recall that for this model, there are 9 estimable parameters (8 survival estimates + 1 reporting probability estimate). In contrast, for the 'equivalent' Brownie model $\{S_t f_.\}$ there are only 8 estimable parameters (7 survival estimates + 1 recovery probability estimate). So, unlike the case where we contrasted the fully time-specific models between the two parameterizations, here, the actual number of estimable parameters differs between the two models. This should suggest fairly strongly that these are not, therefore, equivalent models. And, as such, we should not expect that the estimates of the common parameter survival S should be the same. As we can see after adding the results to the browser

Results Browser: Dead Recoveries

Model	AICc	Delta AICc	AICc Weight	Model Likelihood	No. Par.	Deviance
{Brownie - S(t)f(.)}	61125.7146	0.0000	0.85011	1.0000	8	24.5108
{Seber - S(t)r(.)}	61129.4125	3.6979	0.13381	0.1574	9	26.2085
{Brownie - S(t)f(t)}	61135.0370	9.3224	0.00804	0.0095	15	19.8253
{Seber - S(t)r(t)}	61135.0370	9.3224	0.00804	0.0095	15	19.8253

that models {Seber -$S_t r_.$} and {Brownie -$S_t f_.$} are **not** equivalent; they have different deviances, and different numbers of estimable parameters. If we compare our reconstituted parameter estimates from the Seber model (below) with those from the 'equivalent' Brownie model (shown on the preceding page),

KEDIT - [C:\Documents and Settings\egc\Desktop\mrk3361z.tmp]

File Edit Actions Options Window Help

mcmc

Real Function Parameters of {Seber - S(t)r(.)}

Parameter	Estimate	Standard Error	95% Confidence Interval Lower	Upper
1:S	0.7142337	0.0122936	0.6895424	0.7377083
2:S	0.7132212	0.0095683	0.6941071	0.7316028
3:S	0.7115103	0.0088103	0.6939398	0.7284669
4:S	0.7160142	0.0087768	0.6985020	0.7328982
5:S	0.7109462	0.0092731	0.6924377	0.7287771
6:S	0.7064264	0.0101995	0.6860455	0.7260126
7:S	0.7178115	0.0111038	0.6955484	0.7390574
8:S	0.7122758	0.0125209	0.6871256	0.7361805
9:r	0.2805204	0.0047519	0.2713017	0.2899277

we see that all of the survival estimates differ between the two models. (Note that the reporting probability estimate is fairly close to the true value of 0.286).

Now, in this particular example, you might suspect that the differences in the survival estimates between the two models (Brownie versus Seber) are 'not that big'. In fact, the relative 'closeness' of the estimates in this example owes more to the fact that the simulated data set is very large, and the underlying (generating) model is very simple (no time variation in any of the parameters).

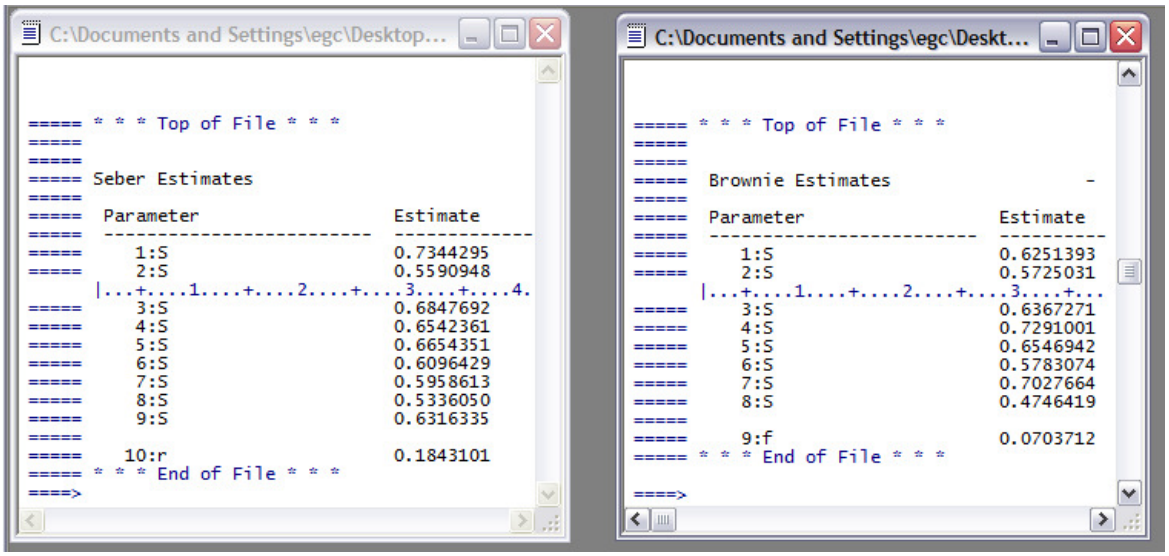
To demonstrate this more graphically, let's reanalyze the recovery data for adult male mallards banded in the San Luis Valley we considered earlier (contained in `BROWNADT.INP`). For these recovery data, fit models {$S_t r_.$} (Seber) and {$S_t f_.$} (Brownie), and add the results to the browser:

Results Browser: Dead Recoveries

Model	AICc	Delta AICc	AICc Weight	Model Likelihood	No. Par.	Deviance
{Seber - S(t)r(.)}	8653.3953	0.0000	0.99733	1.0000	10	44.4485
{Brownie - S(t)f(.)}	8665.2441	11.8488	0.00267	0.0027	9	58.3032

We see that the model fits are not even remotely similar. This is reflected both in terms of the model deviances, but also (and more to the point we're trying to make here) in terms of the parameter estimates.

Here are the reconstituted estimates from the Seber and Brownie models, respectively:



C:\Documents and Settings\egc\Desktop...

```
===== * * * Top of File * * *
=====
=====
===== Seber Estimates
=====
=====  Parameter                  Estimate
=====  -------------------------  -------------
=====     1:S                     0.7344295
=====     2:S                     0.5590948
      |...+....1....+....2....+....3....+....4.
=====     3:S                     0.6847692
=====     4:S                     0.6542361
=====     5:S                     0.6654351
=====     6:S                     0.6096429
=====     7:S                     0.5958613
=====     8:S                     0.5336050
=====     9:S                     0.6316335
=====
=====    10:r                     0.1843101
===== * * * End of File * * *
====>
```

C:\Documents and Settings\egc\Deskt...

```
===== * * * Top of File * * *
=====
=====
=====  Brownie Estimates                -
=====
=====  Parameter                  Estimate
=====  -------------------------  ----------
=====     1:S                     0.6251393
=====     2:S                     0.5725031
      |...+....1....+....2....+....3....+...
=====     3:S                     0.6367271
=====     4:S                     0.7291001
=====     5:S                     0.6546942
=====     6:S                     0.5783074
=====     7:S                     0.7027664
=====     8:S                     0.4746419
=====
=====     9:f                     0.0703712
===== * * * End of File * * *

====>
```

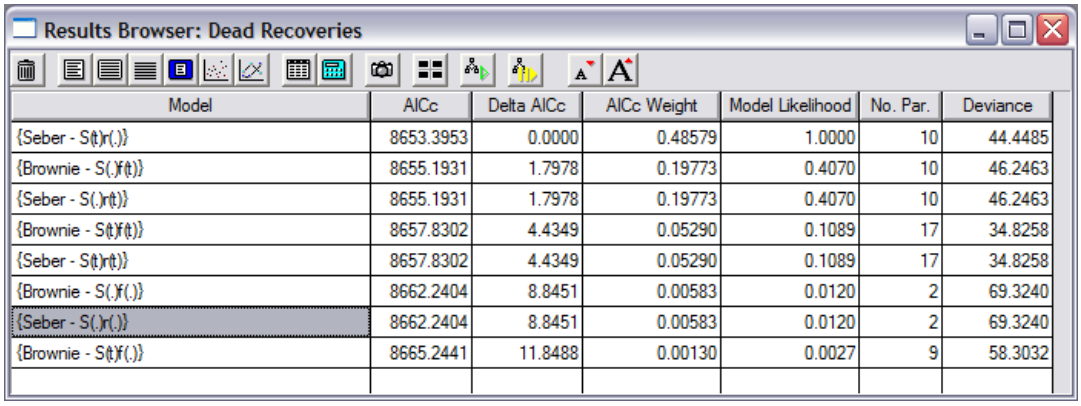
Several things to note. First, there is one more survival parameter estimated for the Seber model, corresponding to the final interval (which is not estimable under the Brownie parameterization).

Second, and of particular note, the estimates of survival for the first 8 intervals which are estimable under both models are quite different - often dramatically so. For example, under the Seber model, the estimated survival probability for the first interval is 0.7344, whereas under the Brownie model, the estimate for the same interval is 0.625, a value which is almost 15% smaller!

So, which model yields estimates of survival which are 'closest to truth'? Well, there are a couple of things to keep in mind. First, for the Brownie model, the recovery probability f contains some information about mortality, and thus constraining either S or f to be constant while allowing the other parameter to vary with time makes implicit assumptions about the pattern of variation in other parameters. For example, for Brownie model $\{S_t f_.\}$, if K (kill rate) varies with time, then parameters c and λ must covary in such a way that the product $Kc\lambda$ (which equals the recovery probability f) does not vary. More likely, if c and λ are constant (as is often assumed), then constant S implies either that K is constant, or that natural mortality is compensatory (and not additive). Thus, it might be reasonable to wonder if model $\{S_t f_.\}$ is a reasonable model under the Brownie parameterization.

Second, and perhaps more practically, it is important to remember that in the end, these are not the same models - the Seber model is more general (i.e., has more parameters) than the Brownie model, and thus will 'fit the data better' (i.e., have a smaller deviance).

And this is key - for most of our models, the Brownie and Seber parameterizations are effectively equivalent - and yield the same model fits and estimates for survival. For example, in the following we show the model fits for our 4 standard models (models $\{S_t f_t\}$, $\{S_. f_t\}$, $\{S_t f_.\}$ and $\{S_. f_.\}$):


Results Browser: Dead Recoveries

Model	AICc	Delta AICc	AICc Weight	Model Likelihood	No. Par.	Deviance
{Seber - S(t)r(.)}	8653.3953	0.0000	0.48579	1.0000	10	44.4485
{Brownie - S(.)f(t)}	8655.1931	1.7978	0.19773	0.4070	10	46.2463
{Seber - S(.)r(t)}	8655.1931	1.7978	0.19773	0.4070	10	46.2463
{Brownie - S(t)f(t)}	8657.8302	4.4349	0.05290	0.1089	17	34.8258
{Seber - S(t)r(t)}	8657.8302	4.4349	0.05290	0.1089	17	34.8258
{Brownie - S(.)f(.)}	8662.2404	8.8451	0.00583	0.0120	2	69.3240
{Seber - S(.)r(.)}	8662.2404	8.8451	0.00583	0.0120	2	69.3240
{Brownie - S(t)f(.)}	8665.2441	11.8488	0.00130	0.0027	9	58.3032

We see clearly that

Brownie		*Seber*
$\{S_t f_t\}$	$\equiv$	$\{S_t r_t\}$
$\{S_. f_t\}$	$\equiv$	$\{S_. r_t\}$
$\{S_. f_.\}$	$\equiv$	$\{S_. r_.\}$
$\{S_t f_.\}$	$\not\equiv$	$\{S_t r_.\}$

In other words, only models $\{S_t f_.\}$ (Brownie) and $\{S_t r_.\}$ (Seber) are not equivalent - because these are the only two models which do not have the same number of estimable parameters. And, thus, comparing survival estimates from these two models is analogous to comparing 'apples and oranges'. We leave the question of which set of estimates (Brownie or Seber) is least biased for you to explore - however, it is clear that you need to pay careful attention to the number of estimable parameters for a given model type if comparing estimates generated using either the Brownie or Seber parameterization.

8.6. Recovery analysis when the number marked is not known

If you look back in this chapter, you'll see that under the 'typical' application of recovery analysis, the number of recoveries expected over a given interval is equal to the number marked and released (N_i) times the survival and recovery probabilities.

However, under some marking schemes (for example, the marking or 'ringing' scheme that was used by the British Trust for Ornithology - the 'BTO'), the number marked and released is often unknown. What can you do in these cases?

To circumvent this problem, a ring recovery model is formulated where the recovery probability (using r_i from the reduced parameterization) is assumed constant by age class and year. Under this assumption, the survival probability can be estimated from the observed recoveries. How does this work?

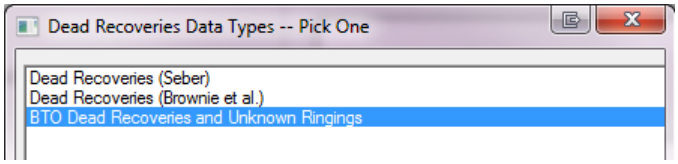
If we assume that r_i is a constant, then the cell probability for the j year of recoveries given k years of recoveries is

$$\frac{S_1 S_2 S_3 \ldots S_{j-1}(1 - S_j)}{1 - S_1 S_2 S_3 \ldots S_k}$$

where the denominator is 1 minus the probability of still being alive. Note in particular that recovery (r_i) does not appear in this expression.

Of the k survival probabilities, only $k - 1$ are identifiable. Common approaches to achieve identifiability are to set $S_{k-1} = S_k$ or to set S_k equal to the mean of $S_1, S_2, \ldots, S_{k-1}$ using appropriate constraints in the design matrix. This model should only be used when you do not know the number of animals marked because you cannot evaluate the assumption of constant recovery probabilities with this model. If you know the number of individuals marked, use one of the 'normal' dead recovery models (Brownie or Seber) described earlier in this chapter.

To implement a BTO recovery analysis, simply select '**BTO Dead Recoveries and Unknown Ringings**':



What about the data file itself? Consider a 'typical' recovery matrix (in fact, the `BROWNADT.INP` file we've looked at previously). The last row of the `INP` file in this case reflect the number released in each year (cohort). What would you do to modify the format for the 'BTO' data type? Simple - delete the last line!

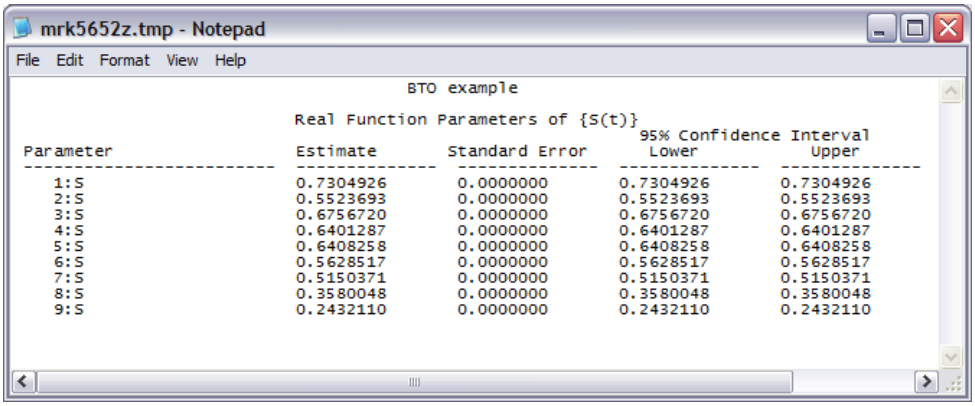
Let's run this analysis using **MARK**, to get a more 'hands-on' sense of how the BTO data type analysis differs from the 'normal' dead recovery analyses we've already discussed. Start up **MARK**, and select `BROWNADT.INP`. View the file, which opens the file in the Windows Notepad. Edit the file by deleting the last row (so that it looks like the above). Save the file from the Notepad - calling it `BTO.INP`. Re-select the file to analyze, this time picking `BTO.INP`. Set the number of occasions to 9, and then make sure the '**BTO Ring Recoveries**' data type is selected.

To see quickly that we're working with something distinct from 'normal' recovery analysis, have a look at the PIM chart. The first thing you'll notice immediately is that there is only one parameter - S (survival). Why? Because recovery (r) is assumed to be constant, and is therefore not estimated. Or, in other words, since the recovery probability (i.e., r_i) does not factor in the expected cell probabilities, then you clearly don't need to estimate it (in fact, you can't!).

With only one parameter, then obviously all constraints are placed on survival only. Clearly, this is a significant limitation in your ability to analyze these data, since you cannot test any hypotheses concerning variation in recovery rate. Assuming a constant recovery rate is a necessary step to do anything with data collected in this way (under the premise that a little knowledge is better than total ignorance). Since the BTO has collected a lot of data over the past many decades, there has been a fair amount of work devoted to the theory of analyzing data of this type, where the number marked and released is unknown. However, there are going to be significant limits to what you can do.

How do the estimates from this analysis, using the BTO data type, compare to those using the 'normal' recovery analysis, where the number marked and released is known (recall that for these data, we actually do know the number marked and released)?

The results from model $\{S_t\}$ for the BTO data type are shown below:


mrk5652z.tmp - Notepad

File Edit Format View Help

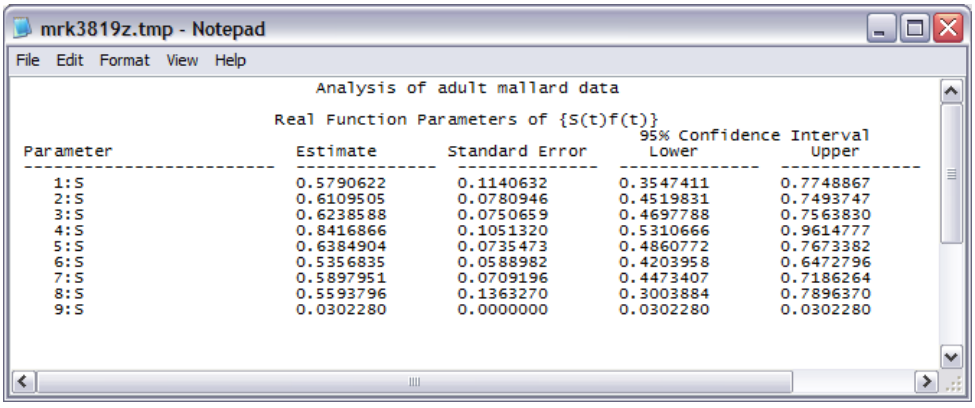
BTO example

Real Function Parameters of {S(t)}

Parameter	Estimate	Standard Error	95% Confidence Interval Lower	Upper
1:S	0.7304926	0.0000000	0.7304926	0.7304926
2:S	0.5523693	0.0000000	0.5523693	0.5523693
3:S	0.6756720	0.0000000	0.6756720	0.6756720
4:S	0.6401287	0.0000000	0.6401287	0.6401287
5:S	0.6408258	0.0000000	0.6408258	0.6408258
6:S	0.5628517	0.0000000	0.5628517	0.5628517
7:S	0.5150371	0.0000000	0.5150371	0.5150371
8:S	0.3580048	0.0000000	0.3580048	0.3580048
9:S	0.2432110	0.0000000	0.2432110	0.2432110

First, notice that there are no standard errors. Error variance around the estimates of S_i cannot itself be estimated under the constraint (assumption) of constant recovery rate.

How do these estimates of S_i compare to the values from the most parsimonious model fit to these data when number marked and released was known? Recall that we analyzed these data earlier in this chapter – referring back to that analysis, we see that the most parsimonious model was model $\{S\,f_t\}$. For this model, S was estimated as 0.638. For the most parsimonious model with time dependence in S (model $S_t f_t$), the estimates of survival are


mrk3819z.tmp - Notepad

File Edit Format View Help

Analysis of adult mallard data

Real Function Parameters of {S(t)f(t)}

Parameter	Estimate	Standard Error	95% Confidence Interval Lower	Upper
1:S	0.5790622	0.1140632	0.3547411	0.7748867
2:S	0.6109505	0.0780946	0.4519831	0.7493747
3:S	0.6238588	0.0750659	0.4697788	0.7563830
4:S	0.8416866	0.1051320	0.5310666	0.9614777
5:S	0.6384904	0.0735473	0.4860772	0.7673382
6:S	0.5356835	0.0588982	0.4203958	0.6472796
7:S	0.5897951	0.0709196	0.4473407	0.7186264
8:S	0.5593796	0.1363270	0.3003884	0.7896370
9:S	0.0302280	0.0000000	0.0302280	0.0302280

We quickly see that the estimates are markedly different. Why? Because, as it turns out, the most parsimonious model(s) had time-dependence in the recovery parameter - clearly a 'violation' of the assumption of constant recovery probability required by the BTO data type analysis.

8.7. Recovery models and GOF

First the good news – you can do GOF testing for recovery models! Now the bad news (well, perhaps not 'bad' news, but something to note) – if you want to use the bootstrap approach for GOF testing, you cannot use the classic Brownie parameterization, but must use alternative '*S* and *r*' (Seber) parameterization. However, if you *are* using the Brownie parameterization, there is another possibility -

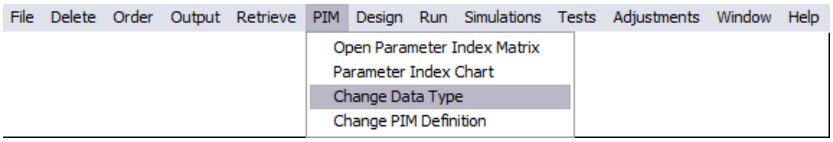
using program **ESTIMATE** to derive an estimate of fit. Program **ESTIMATE** can be called from within **MARK** (much as you can invoke program **RELEASE** from within **MARK**).

Let's start by using a bootstrap approach. We'll use the `BROWNADT.INP` data file we analyzed earlier in the chapter. Recall that the results of our analysis for these data using the default $\hat{c}$ of 1.0 were:

Results Browser: Brownie et al. Dead Recoveries

Model	AICc	Delta AICc	AICc Weight	Model Likelihood	No. Par.	Deviance
{S(.) f(t) PIM}	8655.1931	0.0000	0.76712	1.0000	10	46.2463
{S(t) f(t) PIM}	8657.8302	2.6371	0.20522	0.2675	17	34.8258
{S(.) f(.) PIM}	8662.2404	7.0473	0.02262	0.0295	2	69.3240
{S(t) f(.) PIM}	8665.2441	10.0510	0.00504	0.0066	9	58.3032

Remember we want to derive the measure of fit (estimate of $\hat{c}$) for our general model, which in this case is model $\{S_t f_t\}$. Now, since we can't use the bootstrap for the Brownie parameterization, do we need to 'start over' - re-doing the analysis under the alternative 'S and r' (Seber) parameterization? Nope - thanks to a slick feature of **MARK**. All we need to do is change the data type. First, make model $\{S_t f_t\}$ active by right-clicking on it in the results browser and retrieving it. Then, pull down the '**PIM**' menu, and select '**Change data type**'.



MARK will present you with a selection of data types which are consistent with the data contained in the PIM. In this case, there are only two such data types: the '**Dead recoveries (Seber)**' (i.e., the S and r Seber parameterization, and the '**Dead recoveries (Brownie et al.)**' (our current data type). We want to switch to the Seber 'S and r' data type, so pick the '**Dead recoveries (Seber)**' option from the list. You won't see anything happen, but you'll now be able to run a model under the 'S and r' parameterization. The model we want to run is model $\{S_t r_t\}$, which is equivalent to model $\{S_t f_t\}$. If you want to check to see that the underlying parameterization has now changed to 'S and r', look at the PIMs.

Once you're sure, go ahead and run the model, and call it model '`S(t)r(t)`'. Add the results to the browser. Note that the AIC, deviance and the number of parameters are identical to that reported for model $\{S_t f_t\}$. Now, all you need to do is run a bootstrap or median-$\hat{c}$ GOF on model $\{S_t r_t\}$. The mechanics are identical to what we described earlier in Chapter 5.

Based on 100 bootstraps, we found that approximately 25% of the bootstrapped deviances were greater than the observed deviance for model $\{S_t r_t\}$, indicating adequate fit. Our estimate of $\hat{c}$ was 1.153, consistent with this result. Now, before applying the $\hat{c}$ adjustment to our results, we first delete model $\{S_t r_t\}$ from the browser, since it is effectively redundant to model $\{S_t f_t\}$. Delete model $\{S_t r_t\}$, and apply the $\hat{c}$ adjustment (using $\hat{c}$ = 1.153).

Since $\hat{c}$ is quite close to 1.0, we anticipate that the model rankings and relative degrees of support won't change much. In fact, this is precisely what we observe:

Results Browser: Brownie et al. Dead Recoveries c-hat = 1.1530000

Model	QAICc	Delta QAICc	QAICc Weight	Model Likelihood	No. Par.	QDeviance
{S(.) f(t) PIM}	7509.3305	0.0000	0.78347	1.0000	10	40.1096
{S(.) f(.) PIM}	7513.3154	3.9849	0.10683	0.1364	2	60.1249
{S(t) f(t) PIM}	7513.4830	4.1525	0.09825	0.1254	17	30.2045
{S(t) f(.) PIM}	7517.7816	8.4511	0.01145	0.0146	9	50.5665

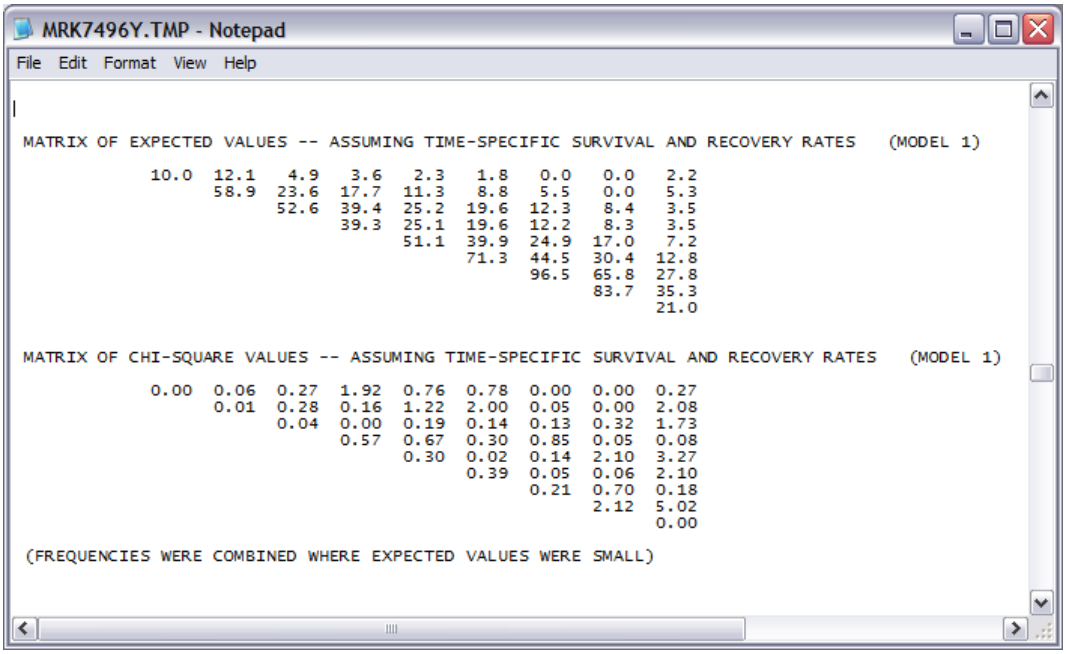
Compare these results to those using $\hat{c}$ = 1.0. Very little difference in this case.

What about program **ESTIMATE** - specifically, what if we don't want to switch to the '*S* and *r*' parameterization for some reason? Program **ESTIMATE** allows you to proceed, because it provides basic GOF testing for several of the 'classic' models under the Brownie parameterization (think of **ESTIMATE** in some senses as the recovery equivalent of **RELEASE**). Program **ESTIMATE** uses the 'classical' naming convention for models we noted earlier in this chapter:

model	*legacy name*	*reference*
$\{S_t f_t\}$	Model 1	Brownie *et al.* (1985) pp. 15-20
$\{S_t f_.\}$	none	
$\{S_. f_t\}$	Model 2	Brownie *et al.* (1985) pp. 20-24
$\{S_. f_.\}$	Model 3	Brownie *et al.* (1985) pp. 24-30

Under this convention, model $\{S_t f_t\}$ is `Model 1`. To run **ESTIMATE**, you don't need to make any particular model in the browser 'active', since **ESTIMATE** simply fits a series of 'built-in' models, regardless of the models you have in your browser. Note: this means that unless your general model is one of the 'built-in' models that estimate is running, you're out of luck. One of these models is `Model 1`. To run **ESTIMATE**, simply pull down the '**`Test`**' menu, and select '**`Program Estimate`**'. After a few seconds, you'll be dumped into the Notepad, which will present the results of the **ESTIMATE** analysis. You'll want to find the part of the output pertaining to `Model 1`. After a bit of searching, you'll find the following results (shown at the top of the next page): the observed χ^2 statistics for model 1 is 31.57, with 25 df. The *P*-value of observing a χ^2-value larger than 31.57 is 17.1%, which is not too far off from the 25% value we had for the deviance using the bootstrap approach (previous page). Further, if we use the model (χ^2/df) as an estimate of $\hat{c}$, then our estimate would be (31.57/25) = 1.263, which isn't too different from our bootstrapped value of 1.153.

So, this would suggest some level of equivalence between the 2 approaches (however, this should be approached cautiously). One thing the **ESTIMATE** output does give you is the *relative* contribution of each element of the recovery matrix to the overall model χ^2. This is analogous to partitioning the data into the contingency tables that we used with program **RELEASE** for live encounter data. The contributions for this data set are shown at the top of the next page. Careful examination of these tables can sometimes help you diagnose lack of fit for recovery data.



8.8. Summary

That's it! Recovery models are more common than you think, and not simply restricted to 'harvested' species. It is worth spending some time getting comfortable with the theory, and the different implementation of recovery analysis in **MARK**. In the next chapter, we'll actually combine 'dead recovery' models with 'live encounter' models. As you'll see, this 'joint' estimation allows you to tease apart sources of apparent mortality in novel and potentially useful ways.

CHAPTER 9

Joint live encounter & dead recovery data

The first chapters in this book focussed on 'typical' open population mark-recapture models, where the probability of an individual being seen was defined by 2 parameters: the probability the animal survived and the probability that an animal alive in state r at time i is alive and in state s at time $i+1$. In Chapter 8, we considered the situation where individuals are 'found dead' (or, recovered), as opposed to encountered alive. In all cases, we model the probability of being encountered (either alive or dead) as a function of underlying parameters. However, up until now, we've only considered what we might call 'either or' models - the marked individuals is either resighted alive, or it isn't. The marked individual is found dead and reported, or it is not. And so forth.

However, clearly there can arise situations where one sort of encounter precludes (or determines or otherwise affects) the probability of an encounter of another kind. The simplest example of this (and the one we will focus on to begin this chapter) is the situation where an individual is found dead. Clearly, if the individual is dead, then it cannot be subsequently resighted as a living individual - the fact that it is dead precludes any other encounter process.

The technical issue (and the major theme of this chapter) is - 'how do we use this extra information in our analysis?' In fact, this 'theme' of 'using extra information' is currently an area of very active research. As we will see, there are many situations in which data of various types (e.g., recoveries, recaptures, telemetry) can be used simultaneously, to provide estimates of parameters that are not estimable using only one source of data, to improve precision of parameter estimates beyond what can be achieved using data from one source only, and to provide some 'flexibility' in accommodating encounter data which might have been collected 'opportunistically' throughout the year. As we will see, the basic ideas for using data from various sources are merely extensions to what we've already discussed. Of course, the 'fun' is in the details.

9.1. Combining live encounters and dead recoveries - first steps. . .

In 1993, Ken Burnham of Colorado State University published a seminal paper outlining an approach for combining dead recovery and live encounter data into a single analysis - we encourage you to read the original text (K. P. Burnham - A theory for combined analysis of ring recovery and recapture data. In '*Marked Individuals in the Study of Bird Population*' (J-D. Lebreton & P. M North Eds) - Birkhäuser-Verlag, Basel). Here, we summarize some of the basic results presented in Burnham (1993), and discuss how the approach is implemented in program **MARK**.

First, we need to identify the basic elements of where the two types of data differ, and where they are

the same. For both dead recoveries, and live encounters, there is some underlying probability that the marked individual will survive over a specified time interval, and that there is some probability that the marked individual will be encountered, either conditional on being alive and in the sample at the end of the interval (in the case of a typical recapture analysis), or dead, recovered and reported during the interval (in the case of a typical dead recovery analysis). Clearly, an individual cannot be recaptured (or otherwise encountered) alive subsequent to being found and reported dead.

However, it is important to remember that the probability of recapture is conditional on (i) being alive, and (ii) remaining in the sampling region (or, as you'll recall from earlier chapters - permanent emigration and mortality are inexorably confounded in a standard recapture study). But what about recoveries? Clearly, it might be reasonable to assume that an individual is equally like to be recovered dead and reported regardless of whether it is in the sample or not. Of course, it is possible under some circumstances that the probability of mortality (recovery) and being reported is dependent on whether or not the marked individual is in or out of the sample area, but for now, we'll assume that the probability of recovery and reporting is independent of sampling location. In other words, we assume that dead recoveries can occur - and be reported - from 'anywhere'.

Given this assumption, then you might have already noted that the addition of recovery data to our analysis gives us something we didn't have before - the ability to separate 'true' mortality from 'apparent' mortality. In simplest terms, in a 'live-encounter' recapture study, the estimate of φ is an estimate of a product of two different events: survival, and fidelity. More formally, we can write

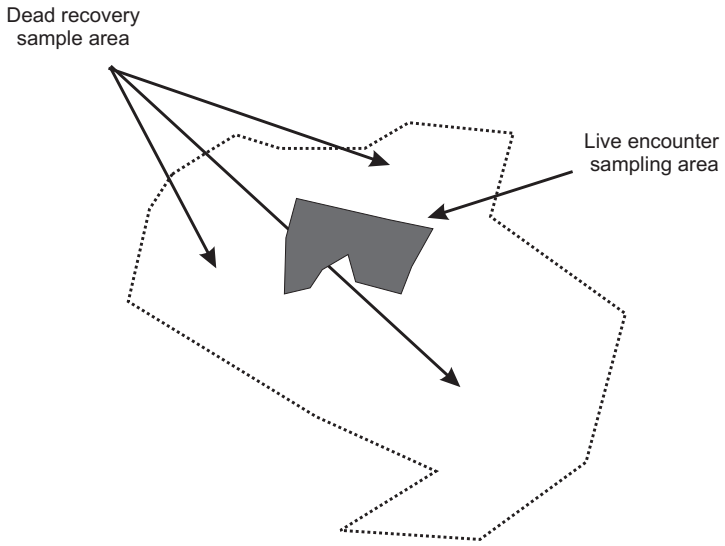
$$\varphi_i = S_i F_i$$

where S_i is the probability of survival from (i) to (i+1), and F_i is the probability of remaining in the sample area (F for 'fidelity') between (i) and (i+1).

However, as we discussed in Chapter 8, S_i can be estimated directly using analysis of data from dead recoveries. As such, we could, in theory (and, as it turns out, in practice) derive an estimate for F_i given estimates for S_i and φ_i (i.e., $\hat{F}_i = \varphi_i / S_i$). Of course, an *ad hoc* way to accomplish this would be to run separate recovery and recapture analyses, and take the estimates from each and 'do some algebra'. However, such an *ad hoc* approach provides no means for estimating the precision of the estimated fidelity parameter, nor the covariance of F_i with the other parameters. Further, the assumption of permanent emigration is not a prerequisite. The 'location' of a live individual during a capture occasion could be random with respect to whether or not it is in the sampling region (and thus at risk of being captured). In this case, the parameter F_i is the probability at time (i) that the individual is at risk of capture given that it is alive, p_i is the probability of capture given that the individual is alive and in the sample (i.e., at risk of capture. What does this mean? Basically, it means that under a 'random' emigration model, φ_i is the true survival probability (S_i), and p_i is the product of F_i and the traditional conditional capture probability. Questions concerning permanent versus temporary emigration, availability for encounter, and so on, are treated more fully in Chapter 15 (which addresses the 'robust design'). Here, we consider one particular approach to separately estimating survival and fidelity.

9.1.1. Estimating fidelity rate, F_i: some key assumptions. . .

The combined 'live encounter-dead recovery' approach originated with considerations of sampling from harvested species. In such cases, the dead recoveries are assumed to occur over a spatial scale that is larger than the scale over which live encounters occur. This is shown in the following diagram.



The dark, smaller area is the area where the marking-live encounters occur. Dead recoveries occur anywhere outside the darker area, but within the area bounded by the dotted line. The key, though, is that the original assumption was that all the dead recoveries occur outside the area where live encounters occur (i.e., outside the dark area, but within the dotted line). If this assumption is met, and if emigration from the sampling area is permanent, then you can partition φ as the product of true survival and fidelity, since in this case, all of the recoveries occur outside of the live encounter sampling area.

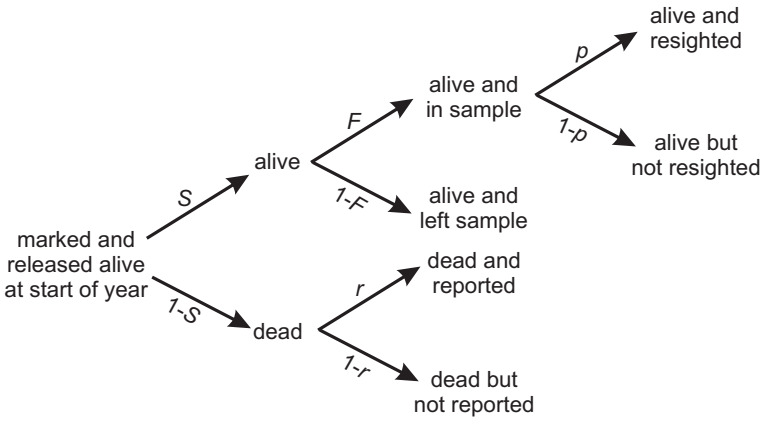
Clearly, this is not always going to be the case - especially for non-game species. In such cases, interpretation of the various parameters is potentially not so simple. If all dead recoveries and live encounters occur in the same sampling area, then clearly realized fidelity F for the marked sample is 1. So, you can use the same approach, except you fix $F = 1$. But, in general, you need to think hard about what the parameters mean, and how the sampling assumptions (and whether or not they are met) influence the interpretation of the parameter estimates.

9.2. Survival, fidelity and encounter probability: probability structure

For now, we'll assume that if an individual emigrates from the sampling area, it is a 'permanent' emigration, such that if the analysis were based entirely on recapture data, an emigrated individual would appear 'dead'. (*Note*: the situation when emigration from the sampling is not permanent, but instead is *temporary*, is considered in Chapter 15, where we develop the 'robust design'). How would such a model be parameterized? Using the 'fate-diagram' graphical approach we have used in previous chapters, consider that fate of a newly marked individual released alive into the population - shown at the top of the next page.

The fate of this individual is governed by several probabilities: S (the probability of surviving the interval), r (the probability of being found dead and reported - the recovery probability using the Seber parametrization discussed in Chapter 8), F (the probability of fidelity to the sampling region - i.e., remaining in the sample. $1 - F$ is the probability of permanently emigrating), and p (the probability of recapture, conditional on being alive and in the sampling region).

Here is the 'fate diagram' for the 'permanent emigration' model. Note that the sequencing of survival before emigration is arbitrary - we could just as easily (and equivalently) place the fidelity 'decision' first. However, the ordering does affect how the probability expressions corresponding to a given encounter history are written.



It is important to keep time-scale in mind when considering this diagram. First, recaptures occur *at* (i), $(i{+}1)...(i+n)$, while recoveries occur *between* (i), $(i+1)...(i+n)$. Second, the number of estimated recapture-based or recovery-based parameters depends on when you end your study. You might recall from Chapter 2 ('data formatting') that joint live-dead analysis uses the '**LD**' encounter history format, where the `L` refers to 'live' and `D` to 'dead' encounters, respectively. Thus, a history of '`1011`' refers to an individual marked and released on the first occasion, that survives the interval between occasion 1 and occasion 2 (since it was subsequently encountered) and then recaptured at occasion 2 and recovered dead during the interval following occasion 2. The '**LD**' format assumes that each recapture occasion is followed by a recovery interval. If your study terminates with the final recapture occasion, and you do not collect recovery data during the following interval, you need to 'code the terminal recovery column (the terminal '**D**' column) as zero, and fix the recovery probability for this interval to 0.

There are several important points to make here. First, the assumption that the terminal live encounter (recapture) occasion is followed by a period (interval) when the individual may be recovered (i.e., encountered dead) means that for standard LD analyses where all parameters are fully time-dependent, the number of parameters for survival (S) and recovery (r) will be one greater than the number of parameters for fidelity (F) and recapture (p). So, the PIMs for survival and recovery parameters will have one more column than will the PIMs for fidelity and recapture. The second point concerns the structural equivalence of the Seber 'S and r' parametrization and the CJS parametrization for live encounter studies (see Chapter 8). As such, you'll need to remember that, as with a CJS analysis of live encounter data, although there are more S parameters than F parameters, the last S is not estimable in a fully time specific model. Its presence is an artifact of modeling recoveries as $(1-S)r$ (using the Seber convention) instead of as f (using the Brownie convention).

Consideration of the probability statements corresponding to various encounter histories will give you a better idea of what is going on. In the table at the top of the next page, we indicate the history in both **LD** format (using **L** and **D** to indicate live encounter or dead recovery, respectively), and the actual binary (0 or 1) coding used in the input file. Each probability statement refers to a different path by which the encounter history could be realized, and assumes that survival occurs before the fidelity 'decision' (as shown in the fate diagram on the preceding page). The total probability of the particular encounter history is the sum of the individual probabilities.

LD history	*binary history*	*probability*
LD00	1100	$(1 - S_1)r_1$
L0L0	1010	$S_1F_1p_2S_2 + S_1F_1p_2(1 - S_2)(1 - r_2)$
L00D	1001	$S_1F_1(1 - p_2)(1 - S_2)r_2 + S_1(1 - F_1)(1 - S_2)r_2$
L0LD	1011	$S_1F_1p_2(1 - S_2)r_2$
L000	1000	$(1 - S_1)(1 - r_1) + S_1F_1(1 - p_2)S_2 + S_1F_1(1 - p_2)(1 - S_2)(1 - r_2)$

Take a moment to make sure you see where the probability statement comes from - they're clearly more involved than those for recapture or recovery analyses alone. For example, consider history '1001' (corresponding to 'L00D'). There are two ways this history could be achieved: the individual clearly survives the first interval (since it is recovered during the second interval, but it could either (1) remain in the sample, and not be seen at occasion 2, and then die and be recovered, or (2) leave the sample area (after which $p = 0$; it cannot be recaptured if it is outside the sample area), and then die and be recovered. The key to remember is that (in theory) a dead recovery can occur whether the individual is in the sample area or not - the same cannot be said for live encounters, which require the marked individual be in the sampling region.

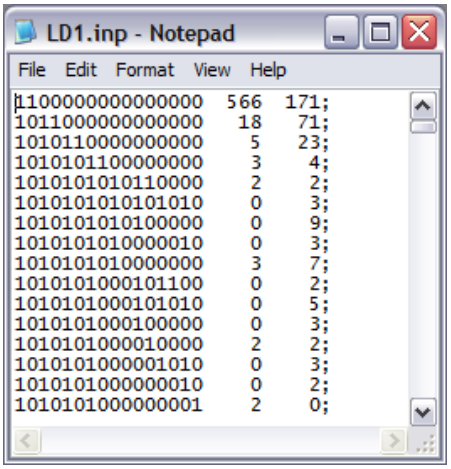
9.3. Combined recapture/recovery analysis in MARK: marked as adult & young

We now address the more practical issue of using program **MARK** to simultaneously analyze dead recovery and live encounter data. To demonstrate the mechanics, we simulated a data set (LD1.INP) with the following structure. We assumed 2 age-classes (young, and adult), with the young age class spanning the first year of life. Within an age class, parameter values were held constant over time - any variation in the parameter values was among age classes. We assume that individuals are marked as both young and as adults at each occasions (recall from Chapter 8 that we suggested that for recovery analyses, we must at least mark adults as well as young. But what about for combined recovery-recapture analyses? More on that later.). The parameter values used in simulating the data set were:

age class	S	p	r	F
young	0.4	0.5	0.6	0.6
adult	0.8	0.5	0.6	0.9

Thus, in the simulated data set, younger individuals had lower survival ($S_y < S_a$), and were less likely to remain in the sampling region during the first year of life (conditional on remaining alive; $F_y < F_a$). Recapture and recovery probabilities were equal for both age classes. There were 8 'occasions' in the simulated data set (where each 'occasion' consists of the recapture event followed by the full year after the recapture event during which recoveries might occur), and 1500 individuals newly marked and released in each age class on each occasion. The simulated data were constructed using the 'decision structure' depicted in the figure shown earlier.

Start **MARK**, and select the data file LD1.INP. We noted already that there were 8 occasions in the data set. However, to remind ourselves about the somewhat different structure of an '**LD**' data set, let's have a look at the INP file (shown at the top of the next page).



Again, given the **LD** data format, for 8 capture occasions, we have 8 corresponding intervals over which recoveries can occur - thus, 16 columns in total. Each consecutive pair of values denotes the encounter history for a given year ('11' - seen and recovered in the same year, '10' - seen but not recovered in a given year, and '01' - not seen but recovered in a given year). Note also that we have two 'groups' individuals marked as young (first frequency column), and marked as adults (second frequency column), respectively. For example, 7 individuals marked as young and 19 individuals marked as adult had encounter history '1010101000000000'.

Next, we need to pick the '**Data Type**'. The third item on the list in the model specification window is '**Joint Live and Dead Encounters (Burnham)**'. This is the one we're after - both recaptures and recoveries, using the approach first described by Ken Burnham. Go ahead and select this data type.

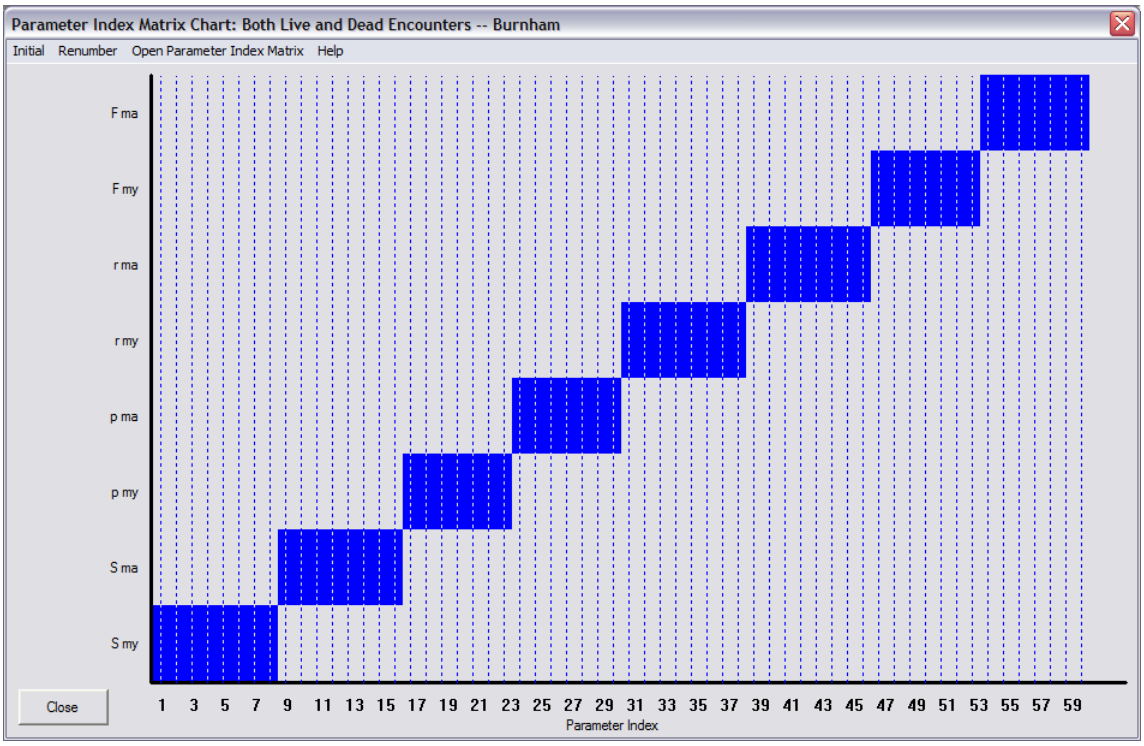
However, before proceeding, one important note: Burnham originally used the 'classic' Brownie parametrization for the 'recovery' side of things - specifically, he used the Brownie parametrization of survival (S) and recovery probability (f). Recall from Chapter 8 that under this parametrization, recovery probability is defined as the product of the kill probability (K), the retrieval probability (c), and the reporting probability (λ) - in other words, f is the probability that the marked individual will be shot (killed), the mark retrieved and then reported. You may also recall from Chapter 8 that **MARK** also allows you to specify a different parametrization (or definition) for recovery probability (r, rather than f), wherein

$$f_i = r_i(1 - S_i)$$

For the joint analysis of dead recovery and live encounter data, **MARK** uses the Seber parameterization (based on parameters 'S and r'), and not the Brownie parametrization originally used by Burnham, primarily to take advantage of increased modeling flexibility this parametrization provides. It is important to keep this in mind.

Once you've confirmed that you've set up the specifications for 8 occasions, two attribute groups (marked as adults and marked as young), and have correctly selected the joint data type (Burnham), click the '**OK**' button to continue. As usual, you're presented with the open PIM for the survival parameter (survival is always the first parameter **MARK** considers). To see the other parameters, open up the PIM chart (shown at the top of the next page). Again, note immediately that the model is specified by 4 parameters (8 'blue' boxes in total - 4 for marked as young, 4 for marked as adult). **MARK** indexes them starting with survival (S), then recapture probability (p), then recovery probability (r), and finally the fidelity parameter (F). As your reading of previous chapters has hopefully made clear, it is essential to

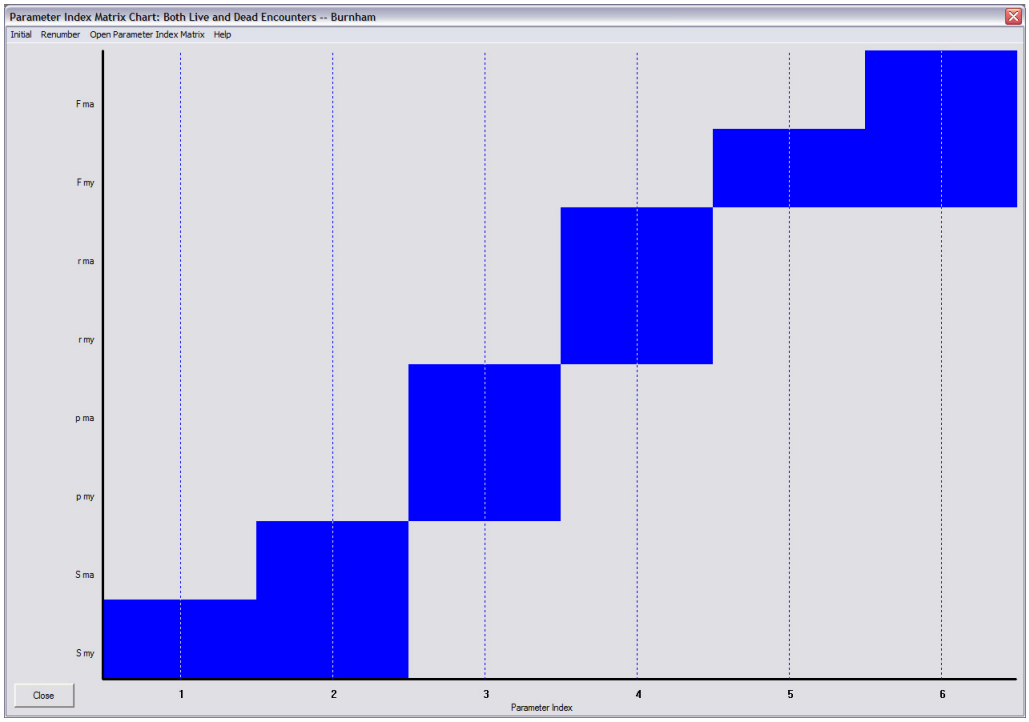
know the sequence that **MARK** uses in 'treating' (indexing) the parameters in the model. Also, confirm that the number of columns in the PIMs for F and p is one less than the number of columns for S and r - make sure you understand why!



Since our purpose here is not to conduct a 'real analysis' (since the data are, after all, completely artificial), we'll only run the 'true' model (under which the data were simulated) - model $\{S_{g-a2/.}p_{.}r_{.}F_{g-.a2/.}\}$. Recall that the quickest way to accomplish this is to modify the PIM chart directly. This is especially true in this case since the 'age structure' we want to impose on the survival, recovery and fidelity parameters is 'constant' (i.e., no temporal variation within age class). This is easily accomplished by right-clicking each of the blue-boxes in the PIM chart (note also that if there had been temporal variation within age class, we'd have to resort to modifying the PIM for each parameter independently, outside the PIM chart - the right-click menu accessible through the PIM chart only allows you to specify a 'constant' age model).

Right-click each of the parameters we want to add age-structure to (S, r and F respectively, for individuals marked as young) , and select 2 as the maximum age (since there are only 2 age classes - young, spanning one year, and adult, spanning the rest of the years in the study). For the parameters corresponding to individuals marked as adults, and for recapture rate for both age classes, we want a constant value - right-click the 'blue-box' for these parameters and select '**constant**'. Also, for recapture, make the recapture probability the same for both age classes by 'stacking' the boxes over each other. Finally, note that survival and fidelity probabilities for 'adults' is the same, regardless of whether the individual was marked as young or adult. Thus, the blue box for 'adult' survival and fidelity should overlap the corresponding parameter in the blue box for individuals marked as you. Then, somewhere in the PIM chart, but not over one of the blue-boxes, right-click and select '**renumber with overlap**' - remember that this eliminates 'gaps' in the parameter indexing, but allows for overlapping for some parameters.

Your PIM chart should look like the following - make sure you see the correspondence between this PIM chart and the true model:



Go ahead and run this model, and add the results to the browser. Here are the parameter estimates for this model (we've added a couple of blank lines to more logically highlight the respective parameters).

mrk1546z.tmp - Notepad

File Edit Format View Help

LD1 - marked young and adult

Real Function Parameters of {test}

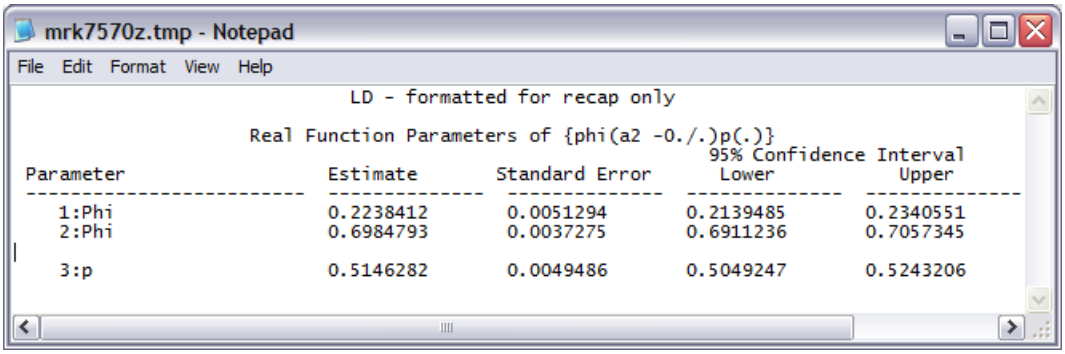
Parameter	Estimate	Standard Error	95% Confidence Interval Lower	Upper
1:S	0.3915571	0.0070583	0.3778133	0.4054751
2:S	0.7949068	0.0040304	0.7868942	0.8026936
3:p	0.5137561	0.0048175	0.5043101	0.5231924
4:r	0.5977567	0.0060602	0.5858234	0.6095753
5:F	0.5773111	0.0138018	0.5500538	0.6041055
6:F	0.8811858	0.0052713	0.8704594	0.8911352

Parameter 1 is the juvenile survival probability (i.e., age 0 $\rightarrow$ 1 year), for individuals marked as juveniles (obviously!). Parameter 2 is the survival probability for adults (including individuals marked as young, and as adults). Both estimates are very close to the 'true' values of 0.4 and 0.8, respectively. Parameters 3 and 4 are the common recapture and recovery probabilities, which are also very close to the 'true' values of 0.5 and 0.6, respectively. Finally, parameters 5-6 are the fidelity probabilities for juveniles and adults, and again, both are close to the 'true' fidelity probabilities of 0.6 and 0.9.

Obviously, the estimates are pretty close to what we expected (not surprising given these are simulated data). However, while this is not particularly noteworthy in its own right, for comparison, let's rerun these same data through **MARK**, using '**Dead recoveries (Seber)**' and '**Live recaptures (CJS)**'. We can do this by editing the `LD1.INP` file accordingly. Can you think of how we would do this? If you can, then you clearly understand the **LD** data format. For the live encounters only analysis, you could either (i) eliminate all the **D** columns, yielding an 8 occasion `INP` file, or (ii) you could simply make all of the **D** columns have a 0 value (i.e., discarding all of the dead recovery data), and use a 'multi-state approach', setting the probability of encounter in the 'dead state' to 0. For this latter approach, we treat dead recoveries as an 'unobservable state' (we cover multi-state models in Chapter 10).

What about for the 'dead recoveries only' analysis? You may recall from Chapter 2 and Chapter 8, there are, in fact, two ways to code recovery data. The 'classic' approach is to use a recovery matrix. However, while traditional, the recovery matrix 'lumps' individuals, and inhibits the ability to constrain estimates as functions of individual covariates (discussed in a later chapter). The encounter history format for recoveries only is also an **LD** format - except that all **L**'s after the initial marking event are coded as 0 (check Chapter 2 to make sure you understand this).

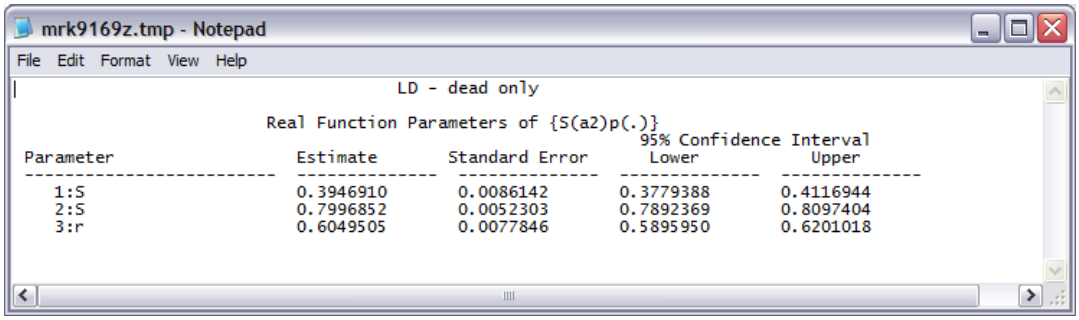
For the live encounters only analysis, we fit model $\{\varphi_{g-a2/.}p_{.}\}$, and for the dead recoveries only model we fit model $\{S_{g-a2/.}r_{.}\}$ - using the non-Brownie (Seber) parametrization. Start with the recaptures only analysis. The results for model $\{\varphi_{g-a2/.}p_{.}\}$ are shown below:



Hmmm. . .are these estimates right? The estimate of $\hat{\varphi}_Y = 0.2238$ doesn't seem particularly close to the value of $S_y = 0.4$ used in the simulation - is there a problem? No! The key is remembering that, under the assumption of permanent emigration, $\varphi_i = S_i F_i$. In the simulation, $F_y = 0.6$, and thus the expectation for $\varphi_y = S_y F_y = (0.4)(0.6) = 0.24$, which is quite close to the estimated value of 0.2238. Similarly, the estimate of $\hat{\varphi}_a = 0.6985$, which is close to the expectation of $\varphi_a = (0.8)(0.9) = 0.72$. The estimate for recapture probability ($\hat{p} = 0.51$) is close to the true parameter value ($p = 0.50$).

Thus, as expected, a recaptures-only analysis provides robust estimates for apparent (or local) survival (φ), and recapture rate. However, since permanent emigration and mortality are confounded in a recaptures-only analysis, the estimate of φ represents only a minimum estimate of true survival, and will generally be lower than the true survival probability (by a factor corresponding to the fidelity rate).

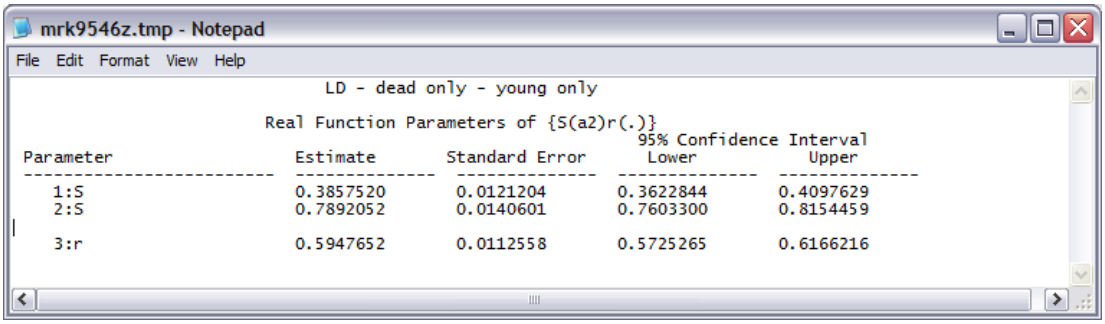
What about the recoveries-only analysis? Clearly, estimates of survival from a recoveries only analysis (shown at the top of the next page) are more accurate estimates of 'true' survival, since they deal directly with dead individuals (i.e., are not confounded by emigration). For this example, where we have marked individuals as both young and adults, we get the following estimates - we see they are essentially identical to the 'true' values for both survival and recovery probabilities.



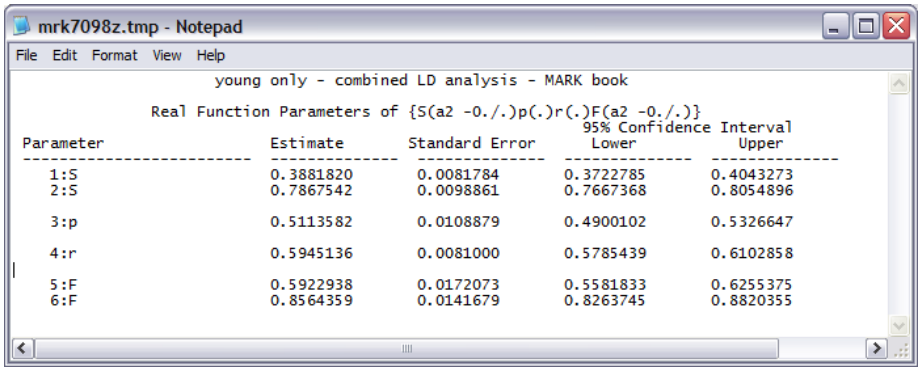
9.4. Marked as young only: combining live encounters & dead recoveries

As we discussed at length in Chapter 8, there are significant difficulties in recovery analysis with data from individuals marked as young only (this is probably a good point to go back and review). Does the 'extra information' from live encounters help us at all?

We can explore this by simply deleting the frequency column for adults from the `LD1.INP` file, and running the analysis. For the recoveries only analysis, using just the data from individuals marked as young, our estimates of survival and recovery probability are:



Not bad, but do we see much improvement if you use the combined live encounter-dead recovery models? The estimates from fitting the 'true' live encounter-dead recovery model to this 'marked as young only' input file are shown at the top of the next page. In this case, the estimates are 'spot on'. This suggests that by combining data from dead recoveries and live recaptures, everything is estimable, with better precision (whereas with a recoveries only analysis from individuals marked as young, everything is not estimable, or if it is, with poorer precision). As noted by Brownie, depending on what assumptions are made concerning constancy of some parameters (as in our example), a few of the parameters in an analysis of recoveries from individuals marked as young are estimable (for example, if we assume constant adult recovery probabilities, adult survival probabilities are estimable, but juvenile survival and both juvenile and adult recovery probabilities are confounded). So, think carefully. But - as this simple example demonstrates, there is great utility in combining data from different sources.



9.5. Joint live-recapture/live resight/tag-recovery model (Barker's Model)

This model extends Burnham's (1993) live/dead model (discussed earlier in this chapter) to the case where live resightings are reported during the open period between live recapture occasions. The population is not assumed to be closed while resightings are being obtained and recoveries of tags from dead animals (dead resightings) may be reported. Because the resighting interval is open it is possible for an animal to be resighted alive several times within an interval then be reported dead. For these animals, only the last dead sighting is used in the model, the earlier live resightings in that period are ignored. Therefore the status of an animal on resighting (live or dead) is determined on the last occasion on which it was resighted in the open interval.

Although the model is complicated and involves 4 sets of nuisance parameters for the recapture and resighting/recovery process, the additional data from resightings and tag recoveries can lead to substantial gains in precision on survival probability estimates. Currently, the model does not allow estimation of abundance or recruitment. For a detailed description of the model see Barker (1997, 1999). Parameters in the model are:

p_i = the probability an animal at risk of capture at i is captured at i

r_i = the probability an animal that dies in $i, i+1$ is found dead and the band reported

R_i = the probability an animal that survives from i to $i+1$ is resighted (alive) some time between i and $i+1$

R'_i = the probability an animal that dies in $i, i+1$ without being found dead is resighted alive in $i, i+1$ before it died

F_i = the probability an animal at risk of capture at i is at risk of capture at $i+1$

F'_i = the probability an animal not at risk of capture at i is at risk of capture at $i+1$ (*NB*: this differs from the definition in Barker, 1997)

The resighting parametrization used in **MARK** differs from that described by Barker (1997). An advantage of the parametrization used by **MARK** is that it enforces certain internal constraints that arise because the joint probability Pr(A and B) should always be less than or equal to the unconditional probabilities Pr(A) and Pr(B). For example, the **MARK** parametrization ensures that the probability an animal is resighted alive in $i, i+1$ and survives from i to $i+1$ is less than the probability it is resighted alive in $i, i+1$. It also ensures that Pr(resighted alive and dies in $i, i+1$ without being reported) < Pr(dies in $i, i+1$ without being reported). These internal constraints are not enforced by the other parametrization.

9.6. Barker Model - Movement

Between trapping sessions, animals are permitted to leave the study area then return. If an animal is in the study area then it is considered 'at risk of capture'. If it leaves the study area it is considered 'not at risk of capture'. Animals that are at risk of capture at time i, leave the study area with probability $(1 - F_i)$. Thus F_i has the same interpretation as in Burnham's (1993) live-dead model as the fidelity to the study area. Animals not at risk of capture are permitted to return to the study area with probability F_i'. In Barker (1997) F_i' was the probability that an animal out of the study area at i remained out of the study area at $i + 1$, but the definition has been changed in the interest of having a parametrization in common with the robust design model.

Under this parametrization there are 3 types of emigration:

Random ($F_i' = F_i$)

Permanent ($F_i' = 0$)

Markov (no constraint.)

A complication is that in the random emigration model the parameters $F_i = F_i'$ are confounded with the capture probability p_{i+1}. By making the constraint $F_i = F_i' = 1$ in **MARK** the random emigration model is fitted, but now the interpretation of p_i is the joint probability that an animal is at risk of capture and is caught, $F_{i-1}p_i$.

Under Markov emigration there tends to be serious confounding of movement and capture probabilities. In a model with time-dependent capture probabilities, it is usually necessary to constrain $F_i = F_.$ and $F_i' = F_.'$ for all i. Even then, the Markov emigration model may perform poorly. In practice the parameters F and F' are usually estimable only if the movement model is markedly different from the random emigration model, that is, if there is a large difference between F_i and F_i'.

To illustrate the meaning of the emigration parameters, suppose the animal is captured during the first trapping session, not captured during the second trapping session, and then captured during the third trapping session. One of several encounter histories that would demonstrate this scenario would be: `100010`

The probability of observing this encounter history can be broken into 4 factors:

P_1 = Pr(animal survives from time 1 to time 3 | released at 1)
P_2 = Pr(animal is not resighted between 1 and 3 | released at 1 and survives to 3)
P_3 = Pr(animal is not captured at 2 but is captured at 3 | released at 1 and survives from 1 to 3 without being resighted)
P_4 = Pr(encounter history after trapping period 3 | events up to trapping period 3)

For describing movement, the relevant factor is P_3. An animal captured at time 1 is known to be at risk of capture at time 1. Because it was captured at time 3 we also know it was at risk of capture at time 3.

There are two possible histories that underlie this observed history:

1. The animal was at risk of capture at time 2 and was not captured, but was captured at time 3
2. The animal left the study area between time 1 and 2 but then returned and was captured.

Because we do not know which one actually occurred we instead find the probability that it was either of the two, which is:

$$P_3 = (1 - F_1)F_2' + F_1(1 - p_2)F_2 p_3$$

The complicated term in the brackets represents the probability that the animal was not captured during the second trapping session but is at risk of capture at time 3. The first product within the brackets $(1-F_1)F_2'$ is the joint probability that the animal emigrated between the first 2 trapping sessions (with probability $1 - F_1$) and then immigrated back onto the study area during the interval between the second and third trapping sessions (with probability F_2'). However, a second possibility exists for why the animal was not captured – it could have remained on the study area and not been captured. The term F_1 represents the probability that it remained on the study area between time 1 and 2 and the term $(1 - p_2)$ is the probability that it was not captured at time 2. The final term F_2 represents the probability that the animal remained on the study area so that it was available for capture during the third trapping session.

Encounter histories for this model are coded as '`LDLDLD`'. Because animals can be encountered in this model as either alive or dead during the interval between capture occasions, 2 different codes are required in the encounter histories to provide information. A '`1`' in the `D` portion of an encounter history means that the animal was reported dead during the interval. A '`2`' in the `D` portion of an encounter history means that the animal was reported alive during the interval. A '`1`' in the `L` portion of an encounter history means that the animal was alive on the study area during a capture occasion.

The following are valid encounter histories for a 5-occasion example: `1010101002`. The animal was captured on the first occasion, and recaptured again on the 2nd, 3rd, and 4th occasions. It was not captured on the 5th occasion, but was seen alive during the last interval.

Consider the history `0000120100`. In this case, the individual was captured on the 3rd occasion, and seen alive during the 3rd interval. It was reported dead during the 4th interval. Note that there can be multiple occasions with a '`1`' in the **L** columns, and multiple occasions with a '`2`' in the `D` columns, but only one **D** column can have a '`1`'.

9.7. Live encounters, dead recoveries & multi-state models

The multi-state model with live and dead encounters is a generalization of the multi-state model that allows inclusion of recoveries of marks from dead animals. Multi-state models are covered in great detail in the next chapter, but we'll introduce some of the basic concepts here.

So what are multi-state models? A simple example will make the basic concepts a bit clearer. Suppose you are conducting a study of some harvested species that is either marked and/or recovered (dead) in any one of three discrete geographical locations. Let the locations (stratum) have the less-than-inspired names of 'region A', 'region B', and 'region C'. Each individual, given that it is alive, will do so on one of these three islands. You are fortunate enough to find sufficient funding so that you are able to mount capture (or resight) operations in all three locations simultaneously. On each occasion, you record encounters with marked individuals, the type of encounter (live, or dead), and in which region the individual was encountered. The information contained in these encounter data allow us to not only estimate survival, but movement/fidelity among regions.

9.7.1. Barker model: assumptions

In addition to the usual assumptions of the multi-state model, this model assumes that apart from group and time effects, the reporting rate of marks from dead animals depends only on the stratum (location) that the animal was in at the immediately preceding live-capture occasion. In some applications, it may be reasonable to also assume that the state of the animal at the time of the dead recovery can be used to determine the state of the animal at the previous live-recapture occasion. This assumption is not included in the model so any such information is ignored.

Model Structure and Likelihood

If there are S strata, define the following:

φ_h is an $S \times S$ matrix with s,t'th element = Pr(animal alive at time h in stratum s is alive at time $h+1$ in stratum t)

ψ_h is an $S \times S$ matrix of transition probabilities with s,t'th element = Pr(animal moves from s to t | alive at h and $h+1$)

P_h is an $(S \times 1)$ matrix with s'th element = Pr(animal alive at time h in stratum s is captured)

S_j is an $(S \times 1)$ vector with s'th element = Pr(animal alive at time j in stratum s is alive at time $j+1$)

r_j is an $(S \times 1)$ vector with s'th element = Pr(animal in stratum s that dies between j and $j+1$ is found and reported)

$\mathbf{D}(x)$ is a diagonal matrix with vector $\mathbf{x}$ along the diagonal, 1 = a $(s \times 1)$ vector of ones

Y_h is an indicator variable that = 1 if the animal was caught at time h, and 0 otherwise.

Note that $\varphi_h = D(Sh)\psi_h$.

The animals in the study can be categorized according to whether their last encounter was as a live recapture or as a dead recovery

Animals last encountered by dead recovery

For an animal first released in stratum s at time i, that was found dead between samples j and $j+1$, and was last captured alive at in stratum t at time k the likelihood, conditional on the first release, is factored into two parts:

1. Pr(encounter history between i and (including) k | first released at time i in stratum s) is the s,t'th element of the matrix formed by taking the product from $h = i$ to $h = k-1$:

$$\prod \psi_h \varphi_h \mathbf{D}(P_{h+1}) + (1 - Y_h)\varphi_h D(1 - P_{h+1})$$

   We take the s,t'th element because we know that the animal was in stratum s at time i and in stratum t at time k.

2. Pr(not caught between k and (including)j and found dead between j and $j+1$ | released at time k in stratum t) is the sum across the t'th row of the matrix formed by taking the product from $h = k$ to $h = j-1$:

$$\prod \varphi_h \mathbf{D}(1 - P_{h+1})\mathbf{D}(1 - S_j)\mathbf{D}(r_j)$$

Although we know that the animal was in stratum t at time k, we do not know which stratum the animal was in at time j. However it must have been in one of the strata and therefore we can find the probability we require by taking the sum across the t'th row of this matrix.

Animals last encountered by live recapture

For an animal first released in stratum s and sample i and last encountered by live-recapture in stratum t and sample j, the likelihood, conditional on the first release, is factored into the two parts:

1. Pr(encounter history between i and (including) j | first released at time i in stratum s) is the s,t'th element of the matrix

$$\varphi_{j-1} D(P_j) \prod Y_h \varphi_h D(P_{h+1}) + (1 - Y_h)\varphi_h D(1 - P_{h+1})$$

   where the product is taken from $h = i$ to $h = j - 2$.

2. Pr(Not encountered again | released alive at j in stratum t). This is found by finding the probability that the animal i encountered at least once after sample j using the above expressions, and then subtracting this probability from 1.

Parameter Identifiability

If the capture occasions are indexed up to sample t and the dead recovery occasions up to sample l, then in addition to the parameters that can be estimated using the multi-strata model, we can also estimate ψ_{t-1}, P_t, S_{t-1} and $r_j (j = 1, ..., t)$. If $l > t$ then (complicated) confounded products of stratum-specific survival and reporting probabilities can also be estimated.

9.8. Summary

One of the recent trends in analysis of data from marked individuals is the increasing focus on using data from a variety of sources. In this chapter, we've looked at simultaneously using data from live encounters and dead recoveries. However, the principles are general - we could also use live encounters combined with known-fate telemetry data, and so on. These developing approaches will increasingly allow us to address several interesting questions which previously were not possible when data from only a single source was used. Next up, multi-state models. . .

CHAPTER 10

Multi-state models...

Many of the first chapters in this book focussed on 'typical' open population mark-recapture models, where the probability of an individual being seen was defined by 2 parameters: the probability the animal survived and remained in the sample area (φ), and the probability that the animal was encountered (p), conditional on being alive and in the sample area.

In this chapter, we extend this simpler paradigm by (in effect) considering a third parameter - a 'movement' parameter (ψ). We'll defer formal definition of this parameter for a moment, since the definition changes somewhat depending on one or more assumptions. However, to foreshadow, let ψ represent the probability of moving between states in which the marked individual may potentially be encountered, conditional on being alive and in that state. The fact that there may be more than a single state (i.e., more than one location, or condition, or state) is what leads to the models we describe in this chapter being referred to generally as *multi-state models*.

Most of this chapter is a synthesis of the basic ideas behind multi-state models, with particular focus on how to implement them in **MARK**. The concepts and ideas are derived from seminal work by Neil Arnason, Carl Schwarz, Cavell Brownie, Ken Pollock, Bill Kendall, Jim Hines and Jim Nichols, who have been exploring the mechanics and application of these models. Critical in this early evolution was the advent of software to fit multi-state models, most notably program **MS-SURVIV**, written by Jim Hines. More recently, the development of programs **M/E-SURGE** by Rémi Choquet, Roger Pradel and Jean-Dominique Lebreton. Multi-state models have been shown to be an extremely rich class of models, with broad applications to many important questions in evolutionary ecology, population dynamics (especially metapopulation dynamics), and conservation biology and management. At best, we hope to provide you with the essence of multi-state models as implemented in **MARK**, sufficient to convince you of the importance of more careful study of this class of models.

So what are multi-state models? A simple example will make the basic concepts a bit clearer. Suppose you are conducting a study of some seabird that breeds on any one of three discrete islands far offshore from any large land mass. Let the islands have the less-than-inspired names of 'Island A', 'Island B', and 'Island C'. Each individual bird, given that it is alive and breeding, will do so on one of these three islands. You are fortunate enough to find sufficient funding so that you are able to mount capture (or resight) operations on all three islands simultaneously. On each occasion (say, each year at the end of the breeding season), you capture, mark and release unmarked individuals, and record recaptures of previously marked individuals. On each occasion, you record the fact that the marked individual was reencountered, and on which island (i.e., the state).

Let's consider what factors will define the probability of encounter. In the 'typical' mark-recapture context, with one sampling location, the probability of encountering the individual in the sample was

© Cooch & White (2014)

defined by the probability that it was alive and in the sampling area (φ), and the probability of encounter conditional on being alive and in the sample area (p).

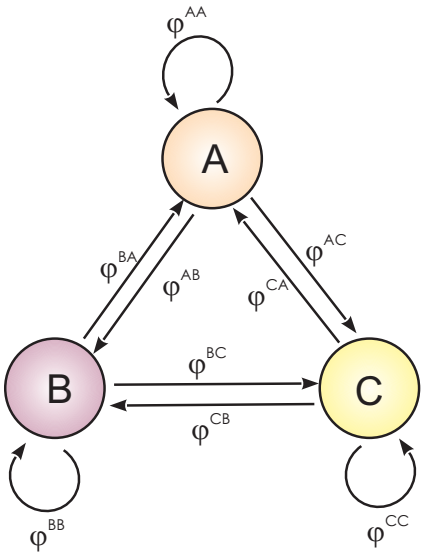
In our 'island' example, though, we have more than one sampling state - we have 3 islands (**A**, **B** and **C**). Suppose you are working on island **B**. You capture an unmarked bird, individually mark and release it. You come back next year, and look for this bird. What determines whether or not you will find it? In effect, a re-reading of the definitions of the parameters for the single-state model provides a clue - the marked bird might be encountered on island **B**, conditional on (a) it surviving to the next occasion, and (b) it not moving to either of the other two islands. As originally described by Arnason (1972, 1973), and later by Brownie *et al*. (1993) and Schwarz *et al*. (1993), the transition probabilities (i.e., making the transition from live to dead, or from one island to another) represent what is known as a *first-order Markov process*. Such a process is defined as one in which the probability of making a given transition between occasion (i) and (i+1) is dependent only on the state at time (i).

Under this assumption, we can now define the parameters which jointly define the probability of encountering a marked individual in a given state on a given occasion:

φ_i^{rs} = the probability that an animal alive in state r at time i is alive and in state s at time i+1

p_i^s = the probability that a marked animal alive in state s at time i is recaptured or resighted at time i.

As written, φ reflects the joint probability of both surviving *and* making a transition. Let's consider this schematically. In the following figure, we show the 3 islands, with arrows indicating the possible transitions.



Remember, the φ values are the probabilities of *both* surviving *and* moving. Now, at this point some of you are probably already leaping ahead to the question '...is there any way to separate these two probabilities - survival and movement?'. We'll come to that in a moment. For now, let's stick with these 2 parameters (φ and p), as defined above. What do our capture (or encounter) histories look like, and what are the associated probability statements? In fact, as discussed briefly in Chapter 2 ('Data Formatting'), the format of the encounter history for multi-state models is qualitatively identical to the 'normal' mark-

recapture history – a contiguous series of variables indicating whether or not the marked individual was encountered on a particular occasion. For 'normal' mark-recapture, this is typically a contiguous series of '1's and '0's.

For multi-state models, instead of '1's to indicate an encounter, we use variables (letters or numbers*) which reflect the particular state in which the individual was encountered. We continue to use '0's to indicate if the individual wasn't encountered in any of the states on a particular occasion.

For example:

encounter history	*interpretation*
AAB0CC	marked on **A** at occasion 1, seen again on **A** at occasion 2, seen on **B** at occasion 3, not seen on any of the islands on occasion 4, seen on **C** at occasion 5 and occasion 6...
BABA00	marked on **B** at occasion 1, seen on **A** at occasion 2, returned to **B** on occasion 3, back to **A** on occasion 4, and not seen on any island at either occasion 5 or occasion 6
ACAAAA	seen on **A** on occasion 1, moved to **C** on occasion 2, then back to **A** and seen on all subsequent occasions in the study

Of course, as we've seen from earlier chapters, each of these encounter histories reflects a particular realization of a probabilistic series of events. It is the relative frequency of each history in the data set which provides the basis for parameter estimation.

Consider a simpler case, with only 2 states: **A** and **B**. What does the encounter history 'AAB' tell us? In this case, the individual was marked and released on **A**, seen again on **A** on the next occasion, and then seen on **B** on the final occasion. What is the corresponding probability expression? Clearly, the organism survived from occasion 1 to occasion 2, and remained on **A**. It also survived from occasion 2 to occasion 3, but in the process, moved from **A** to **B**. Thus, for the encounter history 'AAB', the corresponding probability expression is $\varphi^{AA} p^{A} \varphi^{AB} p^{B}$ (note that for convenience we do not show the subscripts corresponding to the occasions - normally we would do so. The absence of subscripting normally would indicate that the probabilities do not change through time).

What about something slightly more complex - like 'A0B'? In this case, the individual was marked in state **A** on occasion 1, released, not seen in either state **A** or **B** on the second occasion, and then seen again on the third occasion in state **B**. What would the corresponding probability statement look like? In this case, the trick is to realize that there are 2 'probability paths' by which this encounter history could occur:

encounter history	*interpretation*
$\varphi^{AA}(1-p^{A})\varphi^{AB}p^{B}$	survived and stayed in state **A**, but not seen in **A** on occasion 2, survived and moved from **A** to **B** and seen in **B** on occasion 3
$\varphi^{AB}(1-p^{B})\varphi^{BB}p^{B}$	survived and moved from **A** to **B** during first interval, not seen in **B** at occasion 2, stayed in state **B** and seen in **B** on occasion 3

The trick is to realize that (i) the individual clearly survives from occasion 1 to occasion 3 - it is simply 'missed' (not encountered) at occasion 2 in either state, and (ii) since we don't know where the individual

* We hope it is obvious to you that '0' (zero) is not a valid variable to use to indicate a particular state. We hope the reason why is also sufficiently obvious.

was at occasion 2 (i.e., in which state), we must accommodate both possibilities - that it stayed in state **A** (where it was originally marked), or that it moved from **A** to **B** during the first interval. As such, the expected frequency of individuals with encounter history '`A0B`' would be:

$$R_1^A\left[\varphi_1^{AA}(1-p_2^A)\varphi_2^{AB}p_3^B+\varphi_1^{AB}(1-p_2^B)\varphi_2^{BB}p_3^B\right]$$

where R_1^A is the number marked and released in state **A** on occasion 1.

10.1. Separating survival and movement

Now, while the ability to estimate the combined probability of surviving and moving is useful for some purposes, it is ultimately limiting for others. For example, suppose that the states don't consist of physical locations (like islands), but breeding states (say, breeder and non-breeder). There is no shortage of literature on whether or not mortality selection operates on individuals as a function of their breeding status (does 'cost of reproduction' ring a bell, or two?). In a typical analysis of the cost of reproduction, we might want to know (1) is survival dependent upon breeding state, and (2) given that the individual survives, is breeding state at time (*i*) a significant determinant of breeding state at time (*i*+1)? In other words, can we separate 'survival' from 'movement'?

The answer is a qualified 'yes' - qualified, because the separation of 'survival' and 'movement' requires making a particular assumption.

Specifically

> *If we assume that survival from time i to i+1 does not depend on state at time i+1, then we can write*
>
> $$\varphi_i^{rs}=S_i^r\psi_i^{rs}$$
>
> *where (i) S_i^r is the probability of survival from time i to i + 1, given that the individual is in state r at time i , and (ii) ψ^{rs} is the conditional probability that an animal in state r at time i is in state s at time i+1, given that the animal is alive at i+1*

Read it again - slowly. The basic idea is to 'separate' the two events - survival, and moving. Think of it this way - the individual is in state **A**. It survives from (*i*) to (*i*+1) with probability S^A based solely on the fact that it was in state **A** at time (*i*). Then, immediately before (*i*+1), it either moves to another state, or stays, with probability ψ^{Ax} (where *x*=**A**, **B**, or **C** in our example). If the independence assumption is met, then the ordering here (survive and move, or move and survive) is arbitrary.

Now, if we make these assumptions, then the sum of the survival/transition probabilities for a given state is equal to the survival probability for that state. In other words,

$$\sum_s \varphi_i^{r-}=S_i^r$$

Consider the following example – assume there are just two states: *s* and *r*. Since $\varphi^{rs}=S^r\psi^{rs}$ and since $\varphi^{rr}=S^r\psi^{rr}$, then $\sum\varphi^{r-}=S^r\psi^{rr}+S^r\psi^{rs}=S^r\left(\psi^{rr}+\psi^{rs}\right)$. Since $\left(\psi^{rr}+\psi^{rs}\right)=1$, then $\sum\varphi^{r-}=S^r\left(\psi^{rr}+\psi rs\right)=S^r(1)=S^r$. The same logic is true (obviously) for $\sum\varphi^{s-}=S^s$.

If you've followed the earlier chapters on standard mark-recapture approaches, you might be thinking that 'while this is a neat trick, the parameters are probably not separately identifiable'. In fact, they are, because of the constraint that $\sum \psi_i^{r-} = 1$. In other words, the transition (movement) parameters ψ_i^{rs} are conditional on survival – and hence, on being present in the study area. The effect of this constraint is that animals that move out of all the states in the study, i.e., move outside the study area, cause the estimates of survival to be biased in the same sense that 'apparent survival' is estimated. That is, emigration off (or, out of) all the states in the study results in 'apparent survival' being 'true survival' times the probability that the animal remains on the study area.

A simple example will make this clearer. Assume that 3 states are sampled: **A**, **B**, and **C**. As noted earlier, the encounter histories must include the information indicating which state an animal was encountered in. For 5 encounter occasions, a history such as 'BCACC' could result. That is, the animal was initially captured in state **B**, captured in state **C** on the second sampling occasion, captured in state **A** on the third occasion, captured in state **C** on the fourth occasion, and then again in state **C** on the fifth occasion. The cell probability describing this encounter history is

$$\left[S_1^B \psi_1^{BC} p_2^C\right] \times \left[S_2^C \psi_2^{CA} p_3^A\right] \times \left[S_3^A \psi_3^{AC} p_4^C\right] \times \left[S_4^C \left(1 - \psi_4^{CA} - \psi_4^{CB}\right) p_5^C\right]$$

where encounter occasions are separated within the square brackets. Note that for the fourth interval, the probability of *remaining* in state **C** is just 1 minus the sum of the probabilities of *leaving* state **C**. This cell probability demonstrates a key assumption of this model: survival is modeled with the survival probability for the state *where the animal was captured*, and *then* movement to a new state takes place. That is, as implemented in **MARK**, *all mortality takes place before movement*. An animal cannot move to a new state where a different survival probability applies, and then die. If it dies, it must do so on (or, as a function of) the current state. If it lives, then it can move to a new state. This assumption is critical if survival probabilities are different between the states. If survival is constant across states, then the assumption is not important. Biologically, this assumption may not always be reasonable.

begin sidebar

Another assumption for MS models...

Previously, we noted one of the key assumptions which we need to make in order to partition φ into subcomponents S and ψ - specifically, that survival from time i to i+1 does not depend on state at time i+1. Obviously, if this assumption is not met, then the estimates of S and ψ may be strongly biased.

Although the preceding assumption is well-known (and usually mentioned at least once in most papers using MS approaches), there is another assumption which has not received as much attention:

> *MS models assume that all individuals make the transitions at the same time (relative to the start or end of the time interval), or if not, that the distribution of the transitions is known*.

The consequences of violating this assumption are treated in depth in a paper by Joe & Pollock:

Joe, M. & Pollock, K.H. (2002) Separation of survival and movement rates in multi-state tag-return and capture-recapture models. *Journal of Applied Statistics*, **29**, 373-384.

This is a useful paper to read - Joe & Pollock discuss the possible biases caused by violation of this assumption.

end sidebar

10.2. A worked example: cost of breeding analysis

Consider the following example. You are studying a single cohort of individually marked adult deer, for 8 years (i.e., you mark a sample of deer on the first occasion, and then simply follow them for 7 more years). On each occasion, the breeding status of the deer can be determined without error (all individuals can be assigned either breeder or non-breeder status). You want to examine the possibility that survival is influenced by breeding status. This example is analogous to an important paper published by Nichols *et al*. (1994) on estimating breeding proportions and costs of reproduction with capture-recapture data (*Ecology* 1994: 2052-2065). Normally, we might consider breeding status as a 'trait', but clearly this is an annually variable phenotypic trait for most organisms. As such, the classic approach of subdividing the sample along trait-lines and looking for differences among the trait groups will not work here. We need another approach. In fact, the multi-state models are just such an approach - we model the movement between breeding states just as we would model movement among physically discrete states.

To demonstrate the point without the complications of 'messy real world data', we've simulated a data set for this 'virtual deer' (`DEER.INP`), using the following parameter values:

parameter	*value*
$S^{breeder}$	0.7
$S^{non-breeder}$	0.8
$\psi^{breeder \rightarrow non-breeder}$	0.4
$\psi^{non-breeder \rightarrow breeder}$	0.8
$p^{breeder}$	0.7
$p^{non-breeder}$	0.7

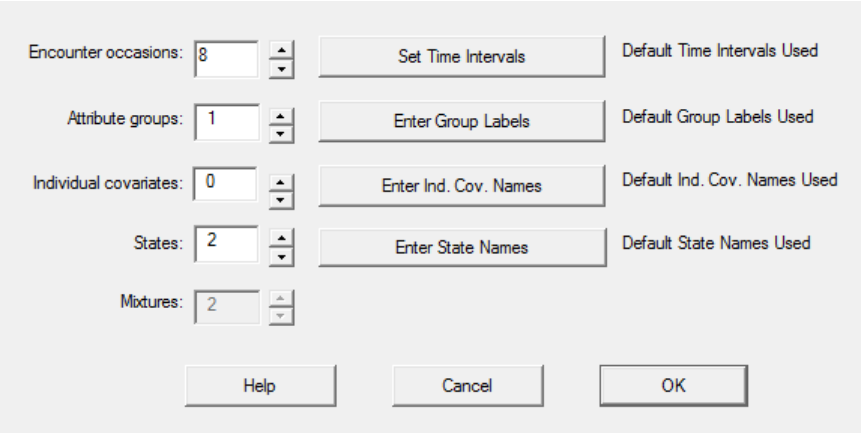
There are 500 total individuals to start in the simulated data - 250 in the breeding state, and 250 in the non-breeding state. If you look carefully at the parameter values, you'll see we're creating a data set where it is 'costly' to breed - the survival from (i) to (i+1) for individuals in the breeding state at (i) is lower than for non-breeders at (i). However, the probability of switching states (moving from one state to the opposite state) is higher for non-breeders than for breeders (i.e., individuals aren't likely to stay non-breeders for very long).

Begin a new project in **MARK**. For the multi-state analysis of the virtual deer, we want to select '**`Multi-strata Recaptures only`**' (about half-way down the list of different data types). Once you select the multi-strata option, you may have noticed that the option to '**`Enter State Names`**' has now become active (lower-right corner of the specification window). This will become important in a minute.

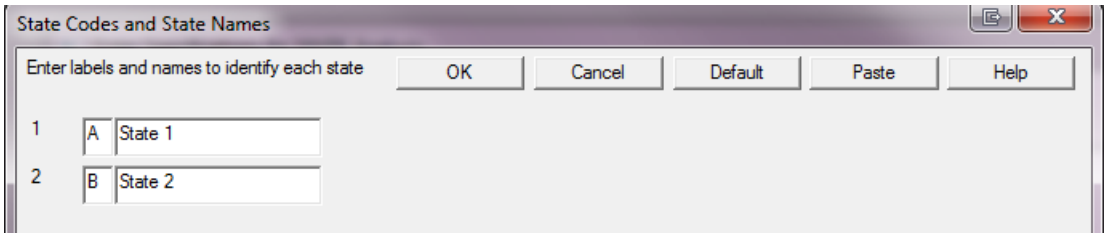
Next, enter a title for the project (say, '`Analysis of virtual deer`'), and then select the file (`DEER.INP`). Note that the encounter histories in `DEER.INP` look virtually identical to the histories we used for typical mark-recapture analysis - a contiguous string 8 characters long (i.e., 8 occasions), followed by the frequency of individuals having that particular history. But remember – rather than '`1`'s and '`0`'s, we now have '`N`'s, '`B`'s and '`0`'s. In this case, the '`N`' value represents individuals in the non-breeding state at a particular occasion, and the '`B`' values are for breeding individuals. The '`0`'s represent occasions when the individual was not seen. Thus, the history '`NBNN0BB`' indicates an individual marked as a non-breeder on the first occasion, seen as a breeder on the second occasion, seen as a non-breeder on occasions 3 to 4, not seen at all on occasion 5, and then seen as a breeder for the final two occasions. The use of '`N`' and '`B`' in this example is entirely arbitrary - you can use anything you want to indicate state (numbers, letters). The only condition is that it can be only one character wide (e.g., '`N`' for non-breeders,

not 'NB').

Next, tell **MARK** that DEER.INP has 8 occasions.* Finally, we need to tell **MARK** how the states are coded in the input file. To do this, click on the '**Enter State Names**' button we referred to earlier (**MARK** defaults to 2 states, so we don't need to change anything there).



This will cause **MARK** to spawn a new window which lets you set the labels (codes) for the different states, and their respective names.

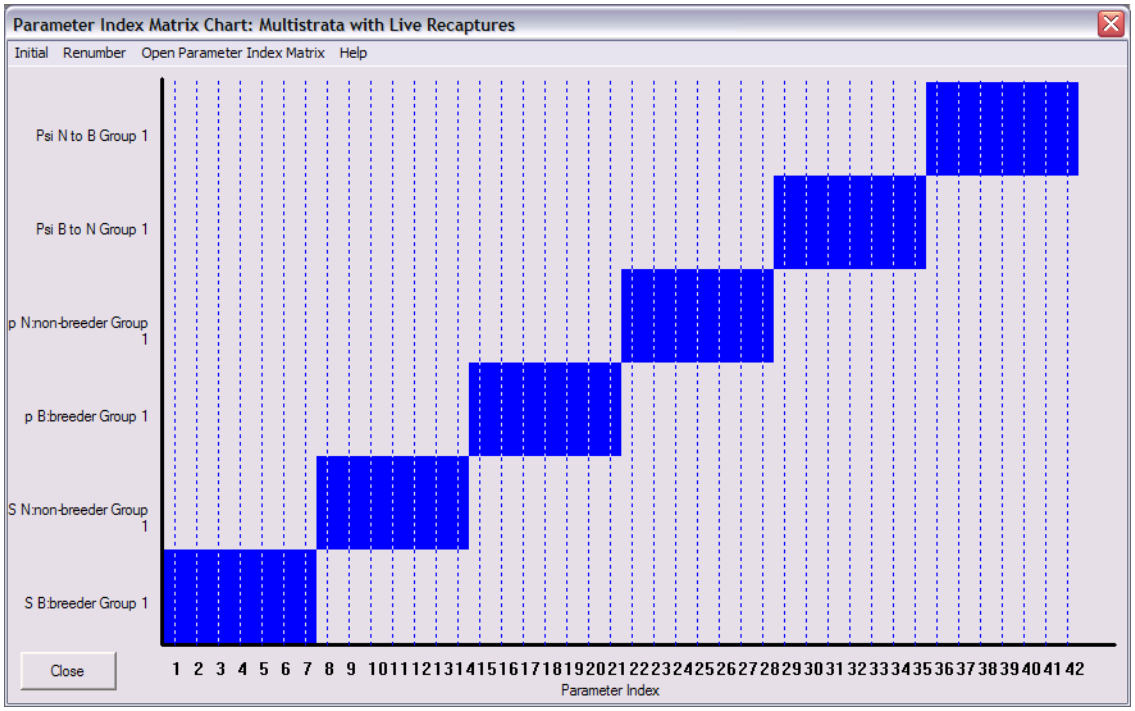


Once you've entered the appropriate codes, and state names, click '**OK**', which will bring you back to the Specification window. Once you're sure everything in this window is correct, click '**OK**'.

As with our standard mark-recapture analyses in **MARK**, what you'll see first is the PIM for the survival parameters for Group 1 (in this example, we have only one group). But, within a group, you might have 2 or more states. If you look at the PIM chart, you should recognize that the PIM reflects a time-dependent structure for survival for individuals in breeding state.

* Here, time intervals are assumed to be equal, 1.0. However, if you have unequal intervals in a multi-state analysis, you need to be very careful. See section 10.6.

To get a quick sense of the way **MARK** lays out the parameters for this model, let's look at the PIM chart (by clicking the '**PIM Chart**' button in the PIM itself):



Clearly, there are 6 parameters involved here. Right away this should tell you something. Six parameters means that **MARK** is making the assumption that survival is dependent only on the state at occasion (i), and is not influenced by the state entered at occasion (i+1). In other words, we're using the identity we introduced earlier.

$$\varphi_i^{rs} = S_i^r \psi_i^{rs}.$$

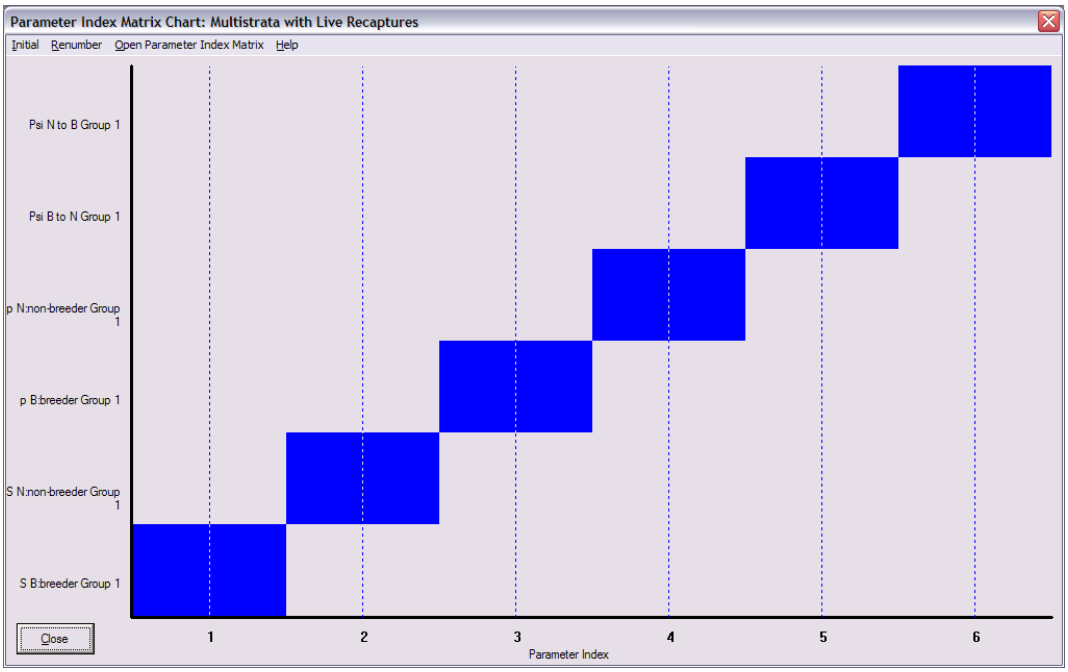
Recall that this identity is true only under this stated assumption. The fact that **MARK** defaults to this assumption becomes important later on. Thus, each of the 'blue boxes' in the PIM chart refers to (respectively, going from the lower-left to the upper-right) S^B, S^N, p^B, p^N, ψ^{BN} and ψ^{NB}. Examination of the horizontal axis of the PIM chart shows that the current model has time-dependence for each parameter, and that there are 42 total parameters. Even though we know that for these simulated data the parameters are constant through time (no time-dependence), let's pretend we're approaching these data naively, and go ahead and run this model (call it '`S(g.t)p(g.t)psi(g.t)`', where the '`g`' refers to group - or state, breeder or non-breeder in this case).

The first thing you might notice, especially if you're using a computer of 'average' processing power, is how much longer this model takes to run than analysis of typical mark-recapture data. The reason is fairly straightforward - the more parameters, the longer it takes to reach the solution (although the increase in time taken does not scale as a simple linear function of the number of parameters). We have 3 parameters (S, p and ψ), so it takes longer than models with only 2 (say, φ and p). Once the estimation is complete, add the results to the browser.

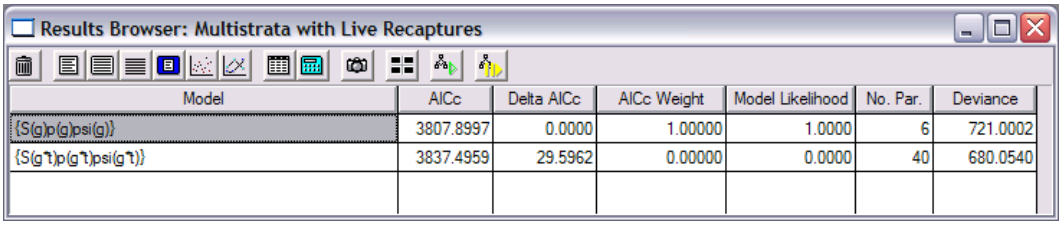
Before we look at the results of this analysis, let's run another model - '`S(g)p(g)psi(g)`' - constancy for all parameters, but allowing for possible differences among groups (breeding states). Obviously,

the first thing we need to do is modify the parameter structure. As you may have gathered from earlier chapters, this is most easily done using the PIM chart. If it isn't open already, go back and open it up. Recall from earlier chapters that we could modify the parameter structure from within the PIM chart (at least for certain models) by simply right clicking on each of the 'blue boxes' on the PIM chart. In this case, you could move the cursor over each of the 'blue boxes' and right-click with the mouse. This causes a menu to pop up which lets you select among various parameter structures. One of the options is '**`Constant`**'. By selecting the '**`Constant`**' option for each blue box in turn, we could build our model. Then, to eliminate the 'gaps' between the 'blue boxes' (i.e., to make the parameter index values contiguous), you could either drag each box manually, or right-click anywhere in the PIM chart, and select '**`Renumber no overlap`**'. This will cause the PIM chart to reformat without any gaps between any of the blue boxes.

While this works, in this case, for this particular model (where the structure is the same for all 6 blue boxes), there is a much faster way - simply select the '**`Initial | All | Constant`**' menu option from within the PIM chart. If you do this, your PIM chart will be quickly reformatted to the model we're after, which looks like the following

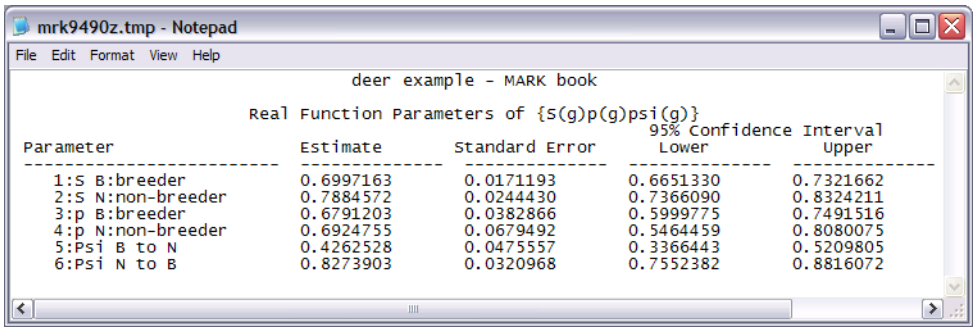


Note that superficially this looks identical to the PIM chart we saw for the fully time-dependent model discussed earlier, but if you look carefully at the horizontal axis, you'll see it now has many fewer parameters - 6 (instead of 42). Go ahead and run this model, and add the results to the browser.



Model	AICc	Delta AICc	AICc Weight	Model Likelihood	No. Par.	Deviance
{S(g)p(g)psi(g)}	3807.8997	0.0000	1.00000	1.0000	6	721.0002
{S(g*t)p(g*t)psi(g*t)}	3837.4959	29.5962	0.00000	0.0000	40	680.0540

Inspecting the results browser shows us immediately that the model with constant parameter values is a much more parsimonious model of these data than is the fully time-dependent model. Before we go much further, let's have a look at the real parameter estimates for the constant model - '`S(g)p(g)psi(g)`' - more formally, model $\{S_g p_g \psi_g\}$.

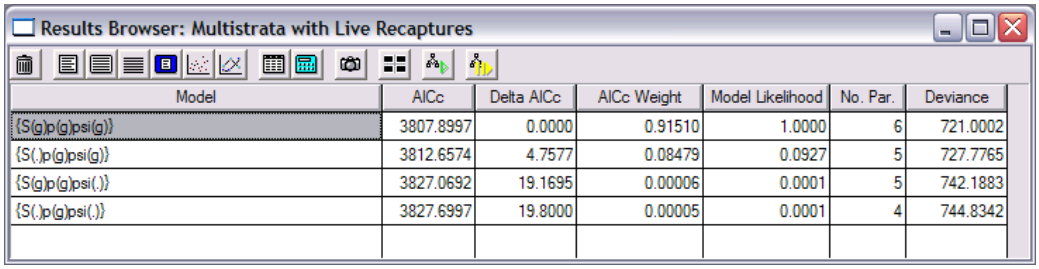


We can see that the estimates are quite close to the parameters used to simulate the data set. Since the constant model is clearly much better supported than the fully-time-dependent model, let's delete the time-dependent results from the browser (by highlighting the time-dependent model and then clicking on the trash can icon in the toolbar of the browser window). We will use the constant model $\{S_g p_g \psi_g\}$ as the general model in our candidate model set. We're interested in examining two questions: (1) is there a difference in survival among breeders and non-breeders, and (2) does the probability of transition between breeding states depend on breeding state at occasion (i)? Let the following represent the candidate model set:

$$\{S_g p_g \psi_g\} \quad \{S p_g \psi_g\} \quad \{S_g p_g \psi\} \quad \{S p_g \psi\}$$

In other words, (1) constant over time, but with group (g; breeding state) differences for all parameters, (2) equal survival between breeding states, but differences in recapture and movement, (3) differences in survival and recapture, but no differences in transitions probabilities between breeding states, and (4) differences in recapture probability among breeding states only.

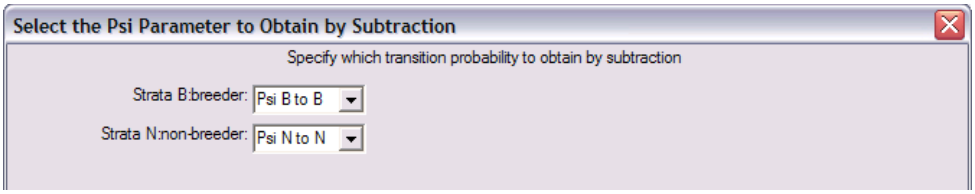
At this point, you should be able to run the 3 new models fairly easily - it should take you only a few seconds to construct each model using the PIM chart approach. The results for all 4 models in the candidate model set are shown at the top of the next page. We see that the model with differences in both survival and movement rates between breeders and non-breeders is 10-times better supported by the data than the next best model, where survival is the same between breeding states, and both recapture and movement probability differ (0.915/0.085 = 10.8).



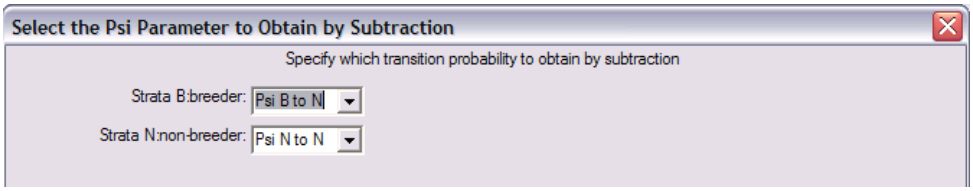
Model	AICc	Delta AICc	AICc Weight	Model Likelihood	No. Par.	Deviance
{S(g)p(g)psi(g)}	3807.8997	0.0000	0.91510	1.0000	6	721.0002
{S(.)p(g)psi(g)}	3812.6574	4.7577	0.08479	0.0927	5	727.7765
{S(g)p(g)psi(.)}	3827.0692	19.1695	0.00006	0.0001	5	742.1883
{S(.)p(g)psi(.)}	3827.6997	19.8000	0.00005	0.0001	4	744.8342

Let's re-examine the estimates from our best model, $\{S_g p_g \psi_g\}$. While survival is clearly estimated for each state separately, the question is, which 'movement' parameter gets estimated? For example, among individuals in breeding state (B) at time (i), they can either move to the other state (with probability ψ^{BN}), or stay in the breeding state (ψ^{BB}). Which one do we estimate? Well, at this point, we take advantage of the logical necessity that the sum of ψ^{BN} and ψ^{BB} must equal 1.0 (i.e., an animal in a given state, must either remain in that state or move to another state, conditional on remaining alive). As such, one of the movement parameters is redundant (if you know the value of one, you know the other as 1 minus the first one). As such, any one of the movement parameters for a given state could be omitted. For example, our estimate for $\psi^{NB} = 0.8274$. Thus, ψ^{NN} is estimated as $\hat{\psi}^{NN} = (1 - 0.8274) = 0.1726$, which is fairly close to the parameter used in the simulation (0.2).

Fortunately, **MARK** gives you some flexibility as to what parameters are estimated - the default is to estimate the probability of moving from one state to another (i.e., ψ^{ij}; probability of moving from i to j). However, you might instead want to estimate the probability of remaining within a state (ψ^{ii}; probability of moving from i to i). You can 'tell' **MARK** to estimate ψ^{ii} simply by changing the definition of the PIM. Returning back to our previous deer example, we can simply retrieve a model from the browser, and then select '**`Change PIM definition`**' from the PIM menu, and run it (or you can change the PIM structure before running the model). Go ahead and try it - doing so will bring up a window showing you the non-default transitions that are available - in this case, there is only one other possibility (i.e., the non-default ψ^{ii}):



Now, you need to be a bit careful here - read the text at the top of this window carefully. It's asking you to tell **MARK** which of the transitions you want to estimate by subtraction. For example, **MARK** defaults to estimating the transition ψ^{ij}. So, you'd have to get ψ^{ii} by subtraction. So, if you want **MARK** to estimate ψ^{ii}, you need to tell it you want to estimate ψ^{ij} by subtraction. So, in our example, with two states (**N** and **B**), **MARK** defaults to estimating ψ^{NB} and ψ^{BN}. So, if you want **MARK** to estimate (for example) ψ^{BB}, you need to tell **MARK** you want to estimate ψ^{BN} by subtraction, as shown below:



If you run this model, you'll see that it gives you the estimate of ψ^{BB} you want. And, changing the specification of which transitions are estimated does not (and should not) change anything else about fitting the model - the model deviance, and AIC value, should be unchanged.

There are several potential advantages to being able to specify which transition parameters are estimated. First, the optimization routine in **MARK** is known to work better if the parameters are not close to the boundaries (i.e., not close to 0.0 or 1.0). Second, you are likely not to want estimates of the estimated parameters to be 'too big', because if their sum is > 1, then the remaining probability is

estimated as < 0. The idea, then, is to pick transition probabilities that are likely to be small, giving you the best chance that the remainder transition probability will be > 0 (although as we will see, we can circumvent this particular problem by specifying a different link function - the multinomial logit link). Third, being able to specify which movement parameter you want to use gives you the ability to build specific constrained models. For example, suppose you are working with a 3-island system, and wanted to assess whether the probability of returning to a given island (i.e., philopatry) was equal for the various islands, but did not want to assume that the probabilities of movement to other islands was also equal. You could do this by constraining the probability of remaining on a given island. Note that for a 2-state (island) model, setting the probabilities of leaving for the other state equal is also setting the probability of remaining equal. With > 2 states (islands), this is not true.

What about φ^{rs}? Recall that as originally described, the multi-state models focussed on estimation of the 'combined' probability of survival and movement. **MARK** assumes that survival is dependent only on state at time (*i*) - this allows **MARK** to separate φ into its component parts S and ψ. Can we estimate φ using **MARK**? Yes, but only by hand. Consider the results of our analysis so far. ψ^{BN} is estimated as 0.4263. S^B is estimated as 0.6997. Thus, φ^{BN} is estimated as $\hat{\varphi}^{BN} = \hat{S}^B\hat{\psi}^{BN} = 0.2983$. Given the standard errors for both S^B and ψ^{BN} from **MARK**, it is possible to derive standard errors for $\hat{\varphi}^{BN}$ using the Delta method (see Appendix B).

Are there further limitations imposed by **MARK**? In fact, there may be one more, stemming from the underlying 'assumption' **MARK** defaults to - the assumption that survival depends only on state at time (*i*). While this is perhaps reasonable in many 'real world' situations, what if it isn't? Nichols and colleagues have extensively explored models where the transition probabilities depend on state both at time (*i*) and (*i*-1). While the recapture parameters remain the same, they introduced a new transition parameter:

$$\varphi_{i-1,i}^{rst} = \text{the probability that an animal alive in state } r \text{ at time } i\text{-}1 \text{ and state } s \text{ at time } i \text{ is in state } t \text{ at time } i\text{+}1.$$

They referred to this as a '*memory model*' (technically, it is a second order Markov model), suggesting that the 'history' of events experienced by the marked individual leading up to its state at time (*i*) might influence the transition probability between (*i*) and (*i*+1). These 'memory' models were coded into **MS-SURVIV***, and allow for testing hypotheses that the transition φ is first-order Markovian (i.e., dependent only on state at time *i*) versus those in which the transition φ is dependent on the state at both (*i*) and (*i*-1) (i.e., second-order Markovian).

At present, **MARK** is unable to handle 'memory' models, in part since they clearly violate the assumption **MARK** makes that survival is dependent only on state at time (*i*). These models may prove increasingly important tools to explore questions concerning life-history decisions over the lifetime of the organism. Much theory exists suggesting that the optimal decisions at age x (e.g., breed or not, emigrate or not) are likely to reflect the sequence of decisions experienced (or made) at age $< x$. However, such memory models are extremely 'data hungry', and much work remains to be done to develop extensions of such models to relevant biological questions.

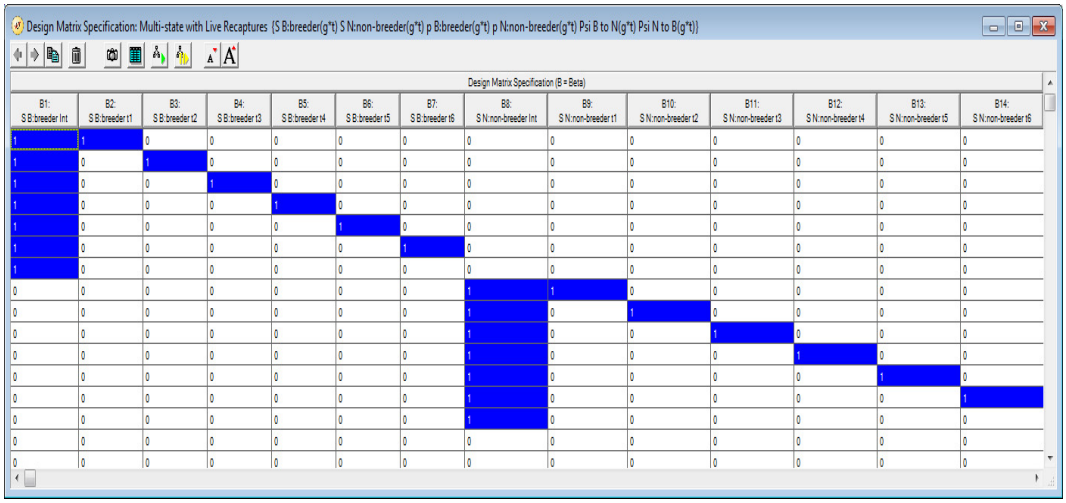
But while this is a limitation of **MARK** when compared to **MS-SURVIV**, for Markovian models, **MARK** adds significant flexibility for many models, particularly through use of the design matrix. In the next section, we shall explore examples showing how we can use the design matrix to constrain the estimates of survival and movement.

* MS-SURVIV is developed and maintained by Jim Hines, USGS-Patuxent Wildlife Research Center

10.3. States as 'groups' - multi-state models and the DM

In the preceding, we fit a series of 'dot' models to the data - there was no need to build a design matrix (DM) for the models in our candidate model set. But, it is important to understand how the DM is built for multi-state models. It is really not much different from what you've already seen – all you need to do is remember that states are, in effect, treated like groups. But, with a catch you need to be aware of.

Consider the deer data we just analyzed, and consider fitting a fully time-dependent model, for all parameters (S, p and ψ). If we set the PIM structure to $\{S_t^g p_t^g \psi_t^g\}$, and then pull up the design matrix using '**`Design matrix | Full`**', this is what we see (if we 'zoom in' on the part of the DM coding for variation survival, S).



If you look closely, you'll see that columns $1 \rightarrow 7$ correspond to survival for breeding individuals, while columns $8 \rightarrow 14$ correspond to survival for non-breeding individuals. What is important to note here is that each parameter has a separate intercept. In other words, **MARK** is treating the same parameter (S) for different levels of the state (breeding, non-breeding) as if in fact they were separate parameters.

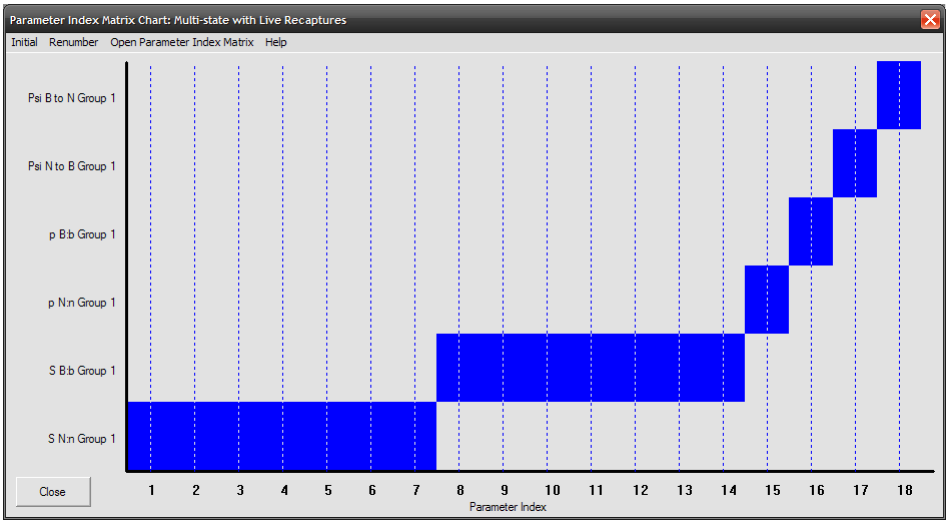
While there is nothing wrong with this in terms of the reconstituted parameter values, it does limit the models you can build. For example, you would not be able to construct an additive model in any obvious way, since there are no *explicit* interaction columns. In fact, the interaction is *implicit* in the fact that the two parameters do not share a common intercept. So, in order to have more flexibility, we generally choose to re-code the DM such that parameters share a common intercept across levels of the state variable. To demonstrate this, we first go ahead and run this model (using the default 'fully time-dependent' DM with a separate intercept for each parameter/state combination. The reported model deviance is 680.054. Now, how would we modify the DM with a common intercept? Simple - we simply have a column for the common `intercept`, a single column for `state` (since there are 2 states, we need only a single column), and then 6 columns for `time`, and 6 additional columns for `state.time` interactions (for a total of 14 columns, identical to the number of columns in the original DM, and equal to the number of parameters specified in the PIMs for survival, S). If you run this modified DM (shown at the top of the next page), the resulting deviance is 680.054, identical to the value reported for the original DM.

Design Matrix Specification: Multi-state with Live Recaptures {ful - default DM}

Design Matrix Specification (B = Beta)

B1 S intercept	B2 State	B3 t1	B4 t2	B5 t3	B6 t4	B7 t5	B8 t6	B9 state*t1	B10 state*t2	B11 state*t3	B12 state*t4	B13 state*t5	B14 state*t6	Parm
1	1	1	0	0	0	0	0	1	0	0	0	0	0	1:S B:breeder
1	1	0	1	0	0	0	0	0	1	0	0	0	0	2:S B:breeder
1	1	0	0	1	0	0	0	0	0	1	0	0	0	3:S B:breeder
1	1	0	0	0	1	0	0	0	0	0	1	0	0	4:S B:breeder
1	1	0	0	0	0	1	0	0	0	0	0	1	0	5:S B:breeder
1	1	0	0	0	0	0	1	0	0	0	0	0	1	6:S B:breeder
1	1	0	0	0	0	0	0	0	0	0	0	0	0	7:S B:breeder
1	0	1	0	0	0	0	0	0	0	0	0	0	0	8:S N:non-breeder
1	0	0	1	0	0	0	0	0	0	0	0	0	0	9:S N:non-breeder
1	0	0	0	1	0	0	0	0	0	0	0	0	0	10:S N:non-breeder
1	0	0	0	0	1	0	0	0	0	0	0	0	0	11:S N:non-breeder
1	0	0	0	0	0	1	0	0	0	0	0	0	0	12:S N:non-breeder
1	0	0	0	0	0	0	1	0	0	0	0	0	0	13:S N:non-breeder
1	0	0	0	0	0	0	0	0	0	0	0	0	0	14:S N:non-breeder

Here is another DM example, again using the deer data. Suppose we decided to fit model $\{S_t^g p_\cdot^g \psi_\cdot^g\}$ - in other words, time varying survival as a function of breeding state (g), with constant encounter and movement probabilities which differ between breeding states. First, build the model using PIMs - here is the PIM chart corresponding to our model:



Run this model, and add the results to the browser. Now, let's build this same model using a design matrix approach. Based on the PIM chart, we see that we have 18 structural parameters in the model. We select '**`Design | Reduced`**', and specify 18 covariate columns in the design matrix. We will follow the convention introduced above, and treat 'state' as a grouping or classification factor. In other words, we'll use a common intercept for modeling differences in survival among states. In this example, we have 2 levels of 'state', so we need one column to code for it.

Here is the design matrix corresponding to our model:

Design Matrix Specification: Multi-state with Live Recaptures

Design Matrix Specification (B = Beta)

B1 int	B2 strata	B3 t1	B4 t2	B5 t3	B6 t4	B7 t5	B8 t6	B9 st1	B10 st2	B11 st3	B12 st4	B13 st5	B14 st6	Parm	B15 pn	B16 pb	B17 psi(N-B)	B18 psi(B-N)
1	1	1	0	0	0	0	0	1	0	0	0	0	0	1:S N:n	0	0	0	0
1	1	0	1	0	0	0	0	0	1	0	0	0	0	2:S N:n	0	0	0	0
1	1	0	0	1	0	0	0	0	0	1	0	0	0	3:S N:n	0	0	0	0
1	1	0	0	0	1	0	0	0	0	0	1	0	0	4:S N:n	0	0	0	0
1	1	0	0	0	0	1	0	0	0	0	0	1	0	5:S N:n	0	0	0	0
1	1	0	0	0	0	0	1	0	0	0	0	0	1	6:S N:n	0	0	0	0
1	1	0	0	0	0	0	0	0	0	0	0	0	0	7:S N:n	0	0	0	0
1	0	1	0	0	0	0	0	0	0	0	0	0	0	8:S B:b	0	0	0	0
1	0	0	1	0	0	0	0	0	0	0	0	0	0	9:S B:b	0	0	0	0
1	0	0	0	1	0	0	0	0	0	0	0	0	0	10:S B:b	0	0	0	0
1	0	0	0	0	1	0	0	0	0	0	0	0	0	11:S B:b	0	0	0	0
1	0	0	0	0	0	1	0	0	0	0	0	0	0	12:S B:b	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0	0	0	0	13:S B:b	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	14:S B:b	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	15:p N:n	1	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	16:p B:b	0	1	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	17:Psi N to B	0	0	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	18:Psi B to N	0	0	0	1

Again, this structure is *identical* to what you have seen before for a single classification factor with 2 levels of the factor. Run this model (label it with DM), and add the results to the browser:

Results Browser: Multi-state with Live Recaptures

Model	AICc	Delta AICc	AICc Weight	Model Likelihood	No. Par.	Deviance	-2Log(L)
{S(g*t)p(g)psi(g) - DM}	3819.0683	0.0000	0.73659	1.0000	17	709.7560	3784.5905
{S(g*t)p(g)psi(g) - PIM}	3821.1249	2.0566	0.26341	0.3576	18	709.7560	3784.5905

Note that the models have different AIC_c values. Is our design matrix wrong? No! Look at the deviances. Note that they are exactly the same. If the deviances are the same, then the models are the same. So, why the difference in the AIC_c values? Remember, the AIC is calculated as a function of the fit (deviance), and the number of parameters. If the AIC values differ, but the fit is the same (i.e., same model deviances), then it is the number of estimated parameters which differ. You see in the browser that this is indeed the case - the model built with the PIM chart shows 18 estimated parameters, whereas the model we just built using the DM reports only 17 parameters. This explains why the two models have different AIC_c values.

begin sidebar

differences in parameter counts

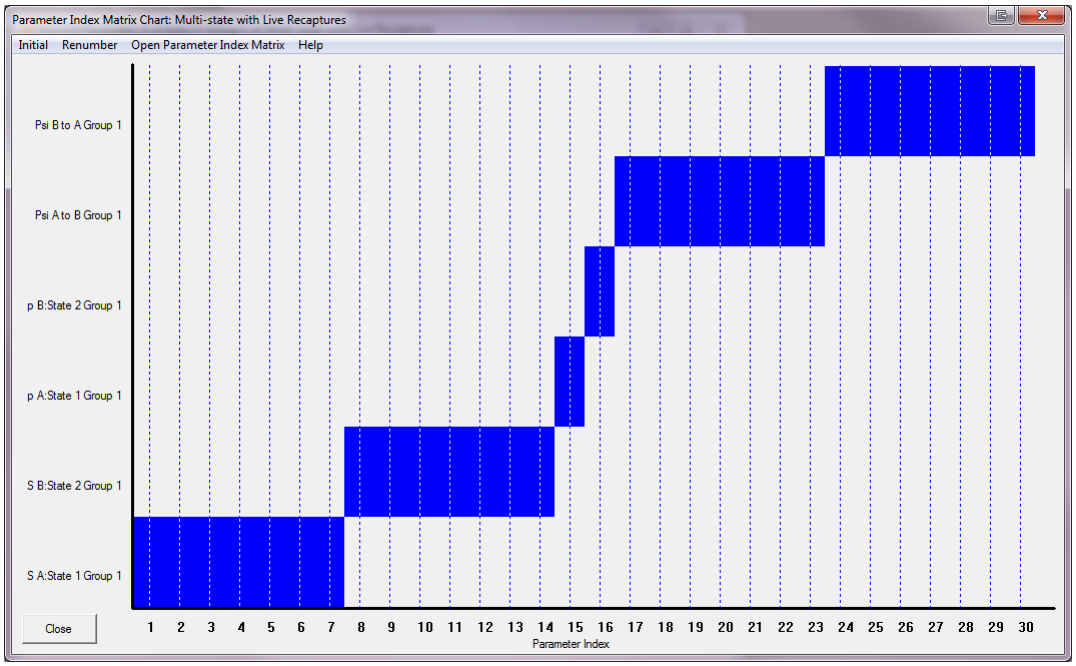
Why the difference between the two models in the number of reported parameters? Again, as discussed in earlier chapters, these differences are often due to differences in the link function used to fit the model to the data. Recall that the default link function for models built using PIMs (and an underlying identity matrix) is the sin link, whereas for models built using the DM, the logit link is used.

Different link functions differ in their ability to resolve parameters - especially those near the $[0,1]$ boundaries. In this case, the logit link (which is known to have some 'difficulty' handling parameters estimated near the boundaries) reported one fewer parameter - if you look at the estimates, you'll see it 'tripped up' trying to estimate survival for the final interval for non-breeding individuals. You can confirm this was indeed the problem by re-running the model you built with PIMs, but first specifying the logit link, instead of the default sin link. If you do so, you'll see that the PIM model run using the logit link reports 17 parameters. Meaning, the problem (difference) is due to the link function, not 'errors' in your DM. Again, recall that in such cases, we would want to manually adjust the number of parameters - in our example, increasing it from 17 to 18. Doing so will make the reported AIC_c values identical between the models built using either PIMs or the DM.

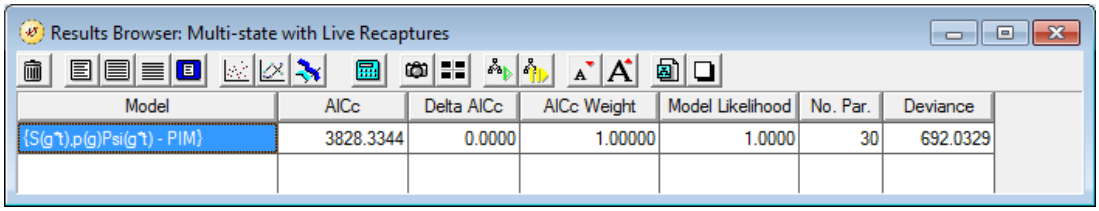
Methods for determining if a parameter is identifiable (given the structure of the model, and the data) are presented in Appendix F.

end sidebar

What about time-varying movement parameters, ψ_i? Let's consider model $\{S_t^g p_\cdot^g \psi_t^g\}$ - time variation in survival and movement, but constant encounter probability. Here is the PIM chart corresponding to this model.



Run this model, and add the results to the browser. We'll call the model '`S(g*t)p(g)Psi(g*t) - PIM`', where the '`PIM`' extension indicates we built this model using PIMs.



We covered the construction of the DM for the survival and encounter parameters for a multi-state design earlier. The DM for these 2 parameters should look like:

Design Matrix Specification (B = Beta)

B1 S-int	B2 S-state	B3 t1	B4 t2	B5 t3	B6 t4	B7 t5	B8 t6	B9 S*t1	B10 S*t2	B11 S*t3	B12 S*t4	B13 S*t5	B14 S*t6	B15 p(B)	B16 P(N)	Parm
1	1	1	0	0	0	0	0	1	0	0	0	0	0	0	0	1:S B:breeding
1	1	0	1	0	0	0	0	0	1	0	0	0	0	0	0	2:S B:breeding
1	1	0	0	1	0	0	0	0	0	1	0	0	0	0	0	3:S B:breeding
1	1	0	0	0	1	0	0	0	0	0	1	0	0	0	0	4:S B:breeding
1	1	0	0	0	0	1	0	0	0	0	0	1	0	0	0	5:S B:breeding
1	1	0	0	0	0	0	1	0	0	0	0	0	1	0	0	6:S B:breeding
1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	7:S B:breeding
1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	8:S N:nonbreeding
1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	9:S N:nonbreeding
1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	10:S N:nonbreeding
1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	11:S N:nonbreeding
1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	12:S N:nonbreeding
1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	13:S N:nonbreeding
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	14:S N:nonbreeding
0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	15:p B:breeding
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	16:p N:nonbreeding

For the movement parameters, ψ, the structure is essentially the same thing:

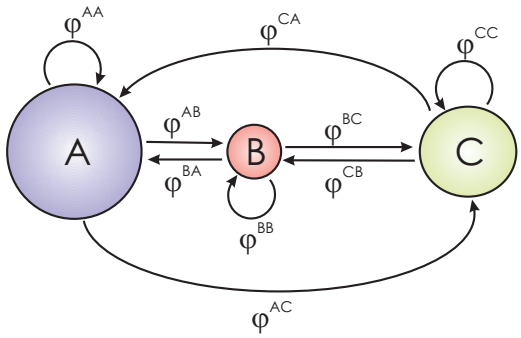
17:Psi B to N	1	1	1	0	0	0	0	0	1	0	0	0	0	0
18:Psi B to N	1	1	0	1	0	0	0	0	0	1	0	0	0	0
19:Psi B to N	1	1	0	0	1	0	0	0	0	0	1	0	0	0
20:Psi B to N	1	1	0	0	0	1	0	0	0	0	0	1	0	0
21:Psi B to N	1	1	0	0	0	0	1	0	0	0	0	0	1	0
22:Psi B to N	1	1	0	0	0	0	0	1	0	0	0	0	0	1
23:Psi B to N	1	1	0	0	0	0	0	0	0	0	0	0	0	0
24:Psi N to B	1	0	1	0	0	0	0	0	0	0	0	0	0	0
25:Psi N to B	1	0	0	1	0	0	0	0	0	0	0	0	0	0
26:Psi N to B	1	0	0	0	1	0	0	0	0	0	0	0	0	0
27:Psi N to B	1	0	0	0	0	1	0	0	0	0	0	0	0	0
28:Psi N to B	1	0	0	0	0	0	1	0	0	0	0	0	0	0
29:Psi N to B	1	0	0	0	0	0	0	1	0	0	0	0	0	0
30:Psi N to B	1	0	0	0	0	0	0	0	0	0	0	0	0	0

Note that is is not necessary to use a common intercept, but it can be convenient to do so for models where you may want to create a structural relationship between the parameters.

10.3.1. A simple metapopulation model - size, distance & quality

In this example, we go back to the hypothetical model we considered right at the beginning - 3 islands with colonies of a particular species of sea-bird, with the potential for exchange among some or all of the islands. For this example, we'll add some complexity to the model, by introducing a number of factors which might potentially influence any of the 3 parameters, either individually or together - factors which are fairly representative of 'real-world' data. In fact, our example is qualitatively quite similar to a recent study of a metapopulation of roseate terns (Spendelow *et al*. 1995 *Ecology* **76**: 2415-2428).

In our example, we'll vary 2 main factors: (1) the size of individual islands, and (2) the spacing (distance) among the islands. Here is a graphical representation of our 'island system':



We see that the islands are clearly not equally spaced: as drawn, island **B** is closer to island **A** than it is to island **C**. And, the islands are not the same size: island **A** is the largest, followed by island **C**, with island **B** the smallest. The islands might differ in terms of some characteristic (say, some limiting resource). Sometimes this might scale with the size of the island. For our example, we'll simplify somewhat: we'll assume that island **A** has the highest 'quality', while island **B** and island **C** have equivalent quality. As indicated, all transitions among islands are possible. The question we have then is - are there differences in survival, movement probability or recapture probability among the 3 islands? Further, might any differences that are apparent correlate with differences in spacing, size or quality of the islands?

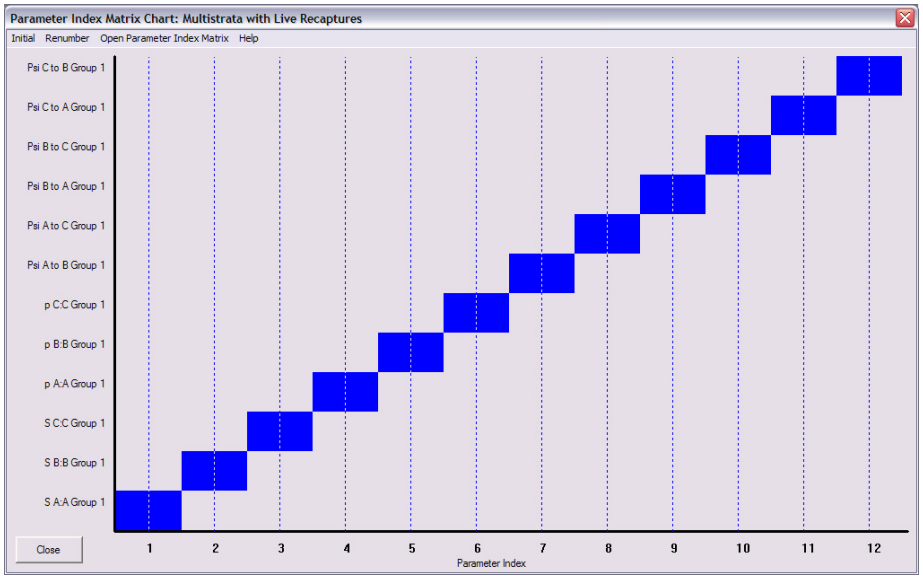
based on this underlying 'metapopulation structure', we simulated a 6 occasion study, with capture, mark and release occurring simultaneously on all three islands in all years. In other words, in this example, we're not simply following a single marked cohort through time, but are releasing recaptures and newly marked birds on each occasion. We simulated 250 newly marked individuals on each island at each occasion. Rather than tell you a priori what the parameter values were in the simulation, let's see how well we can do by building a candidate model set, and using Akaike weights to select the best model. Based on our description of the system, we have good reason to expect that island quality might influence survival. Further, distance among islands, and differences in island size, might influence movement probabilities. Of course, we might also hypothesize that island size could influence both survival and movement if we invoked density-dependent effects (which will tend to lead to departures from the ideal free distribution based on simple differences in quality). For now, let's say that, based on earlier studies of this species, we have no evidence for density-dependent effects. Next, assume there is a fixed number of investigators on each island capturing and releasing the birds. Assume also that, all other things being equal, colony size is proportional to the size of the island. Thus, since island **A** is the largest, you might anticipate that for a constant level of capture effort, that recapture probability should be smaller on island **A** than on (say) island **B**, which is the smallest of the three islands. Finally, we note that environmental conditions during the 6 years of the study have been near-constant (as far as we can tell).

Based on these *a priori* hypotheses, we construct our candidate model set (tabulated on the next page). Given the detail of our background knowledge, and our insight about these birds, we start with a general model which allows for 'group' differences (i.e., differences among islands), but one that is constant over time. We include reduced parameter models starting from this general model. The model set is obviously not exhaustive, nor is it necessarily the most appropriate one (that is at the discretion of the analyst - and lest we forget, these are simulated data after all). However, it is fairly representative of

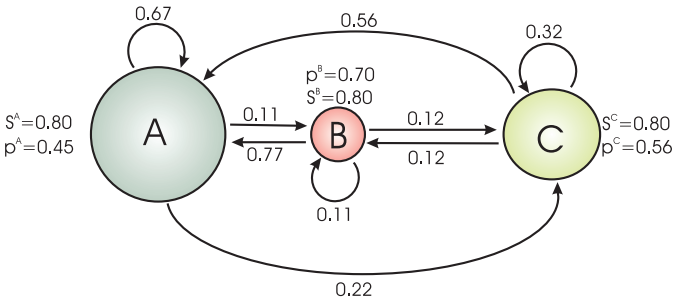
what you might have included if this were a real-world analysis.

model	*description*
$S^g p^g \psi^g$	general model - all groups (states) different - constant over time
$S p^g \psi^g$	constant survival - group difference in recapture and movement
$S^g p \psi^g$	constant recapture - group differences in survival and movement
$S^g p^g \psi$	constant movement (inter-island all equal) - group differences in survival and recapture
$S^{quality} p^g \psi^g$	survival constrained to be a function of island quality - group differences in recapture and movement
$S^g p^{size} \psi^g$	recapture constrained as a function of island size - group differences in survival and movement
$S^g p^g \psi^{distance}$	inter-island movement a function of inter-island distance - group differences in survival and recapture
$S p^{size} \psi^{distance}$	recapture a function of island size, movement a function of inter-island distance - constant survival
$S^{quality} p^{size} \psi^{distance}$	survival a function of island quality, recapture a function if island size, and inter-island movement a function of inter-island distance

We'll start by fitting the general model - $\{S_g p_g \psi_g\}$. The data file containing our simulated data is called `ISLAND.INP`. There are 6 occasions, and 3 states (we used `A`, `B` and `C` as the coding variables in the encounter histories). Label the 3 states as '`island A`', '`island B`' and '`island C`', respectively. As with our first example with the 'virtual deer', we can parameterize this model most easily by modifying the PIM chart. Remember that when we start a new project, **MARK** defaults to the fully time dependent model. Open up the PIM chart, and change the underlying parameter structure to one that has group (i.e., island) differences, but that is constant over time for each parameter. Again, this is most efficiently done by selecting the '**`Initial | All | Constant`**' menu options from within the PIM chart. If you've done it correctly, your PIM chart should now look like the following:



Go ahead and run the model, and add the results to the results browser. The AIC_c for this model is 19703.54, with 12 estimated parameters. Since this is our general model, let's have a preliminary look at the parameter values. To make it easier to relate the estimates to the model, we'll add the estimates to our 'model diagram', shown below:



Clearly, there is some heterogeneity among islands for recapture and movement probability, but not survival. The question is, are the apparent differences in recapture or movement significant, and does the pattern of variation correlate with one or more of the covariates in our model(s)? Since the 2nd through 4th models in the model set (preceding page) are straightforward (hopefully!), we'll skip the mechanics of setting them up and running them, and go ahead and present the results (shown at the top of the next page). The best model $\{Sp^g\psi^g\}$ is not appreciably better supported by the data than is the next best model $\{S^gp^\cdot\psi^g\}$. As such, we could only say that these two models are probably equally likely. So, our tentative conclusion at this point is that there is some evidence of equivalence (in some senses) of survival among islands. The movement probabilities clearly differ. This would seem to be concordant with our casual inspection of the estimates from the most general model (shown in the schematic diagram, above).

Results Browser: Multistrata with Live Recaptures

Model	AICc	Delta AICc	AICc Weight	Model Likelihood	No. Par.	Deviance
{S(.)p(g)psi(g)}	19700.1861	0.0000	0.49366	1.0000	10	1605.7806
{S(g)p(.)psi(g)}	19700.5381	0.3520	0.41399	0.8386	10	1606.1323
{S(g)p(g)psi(g)}	19703.5388	3.3527	0.09234	0.1871	12	1605.1185
{S(g)p(g)psi(.)}	19781.7919	81.6058	0.00000	0.0000	7	1693.4039

What about GOF (goodness of fit)? Normally, at this stage, we'd be thinking about fit of our general model - in part for the purposes of deriving an estimate of $\hat{c}$. We discuss GOF testing for multi-state models at the end of this chapter - for this example, we'll assume the general model fits the data, and leave $\hat{c}$ at the default value of 1.0.

Can we improve our understanding by constraining the general model to be a function of one or more of the 'potentially relevant' covariates? At this point we need to modify the design matrix to constrain the general model. Click once on model $\{S^gp^g\psi^g\}$ in the browser - to make it the active model. Then, right-click this model, and select the '**Retrieve**' option. Since we want to constrain model $\{S^gp^g\psi^g\}$, we want the design matrix to initially reflect its parameter structure.

Before we actually look at the design matrix, what would the linear model look like for the model you just retrieved? Remember, the retrieved model was $\{S^gp^g\psi^g\}$. Since 'group' = 'island', then there are 3

levels of group (i.e., 3 islands). Thus, we need $(3 - 1) = 2$ columns, plus a column for the intercept, to code for the various group effects for survival and recapture. The structure of the design matrix for the S and p parameters is (hopefully) straightforward - a column for the intercept, followed by 2 columns of dummy variables, '1 0' for island **A**, '0 1' for island **B**, and '0 0' for island **C**. For this model, we'll use the same basic linear structure for both parameters (S and p). Remember, these codings are arbitrary - we could have just as easily used '1 0' for **A**, '1 1' for **B**, and '0 1' for **C**. The important point is that what is required is 2 columns for the coding, and that the coding is based on 0 or 1 dummy variables.

What about the ψ (movement) parameters? A little trickier, since there are several equivalent coding schemes which would accomplish the same thing. Note that there are 3 groups of movement parameters - those involving movements from island **A**, those involving movements from island **B**, and those involving movements from island **C** - 2 parameters for each group. Thus, following the standard linear models paradigm, we could also use 1 intercept column, and 1 column to indicate which of the 2 transitions within each of the 3 movement groups. For example, as shown below, for the parameters ψ^{AB} and ψ^{AC}, we could have a column of intercept, followed by a column with a '0' indicating movement from **A** to **B**, and a '1' indicating the **A** to **C** movement.

The design matrix for all 3 parameters is shown below:

Design Matrix Specification: Multistrata with Live Recaptures

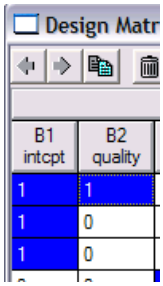
Design Matrix Specification (B = Beta)

B1	B2	B3	B4	B5	B6	Parm	B7	B8	B9	B10	B11	B12
1	1	0	0	0	0	1:S A:A	0	0	0	0	0	0
1	0	1	0	0	0	2:S B:B	0	0	0	0	0	0
1	0	0	0	0	0	3:S C:C	0	0	0	0	0	0
0	0	0	1	1	0	4:p A:A	0	0	0	0	0	0
0	0	0	1	0	1	5:p B:B	0	0	0	0	0	0
0	0	0	1	0	0	6:p C:C	0	0	0	0	0	0
0	0	0	0	0	0	7:Psi A to B	1	0	0	0	0	0
0	0	0	0	0	0	8:Psi A to C	1	1	0	0	0	0
0	0	0	0	0	0	9:Psi B to A	0	0	1	0	0	0
0	0	0	0	0	0	10:Psi B to C	0	0	1	1	0	0
0	0	0	0	0	0	11:Psi C to A	0	0	0	0	1	0
0	0	0	0	0	0	12:Psi C to B	0	0	0	0	1	1

Try running this design matrix – the estimates are identical to the estimates for the model you created initially simply by modifying the PIMs.

Now that we have built our general model using the design matrix, let's build the next model in the set, model $\{S^{quality}p^{g}\psi^{g}\}$. For model $\{S^{quality}p^{g}\psi^{g}\}$, we want to constrain survival to be a function of island 'quality'. Recall that in this example, island **A** is believed to be of better quality than island **B** or **C**, but that island **B** and **C** are believed to be of equal quality. Thus, we need a single column for the intercept, and a single column coding for quality. We'll let '1' represent 'good quality', and '0' represent 'poor quality'.

Thus, the design matrix corresponding to the survival parameters would look like:



Go ahead and run this model, and add the results to the browser. Before we examine the results of this model, let's go ahead and fit the next 2 models in the list - model $\{S^g p^{size} \psi^g\}$ and model $\{S^g p^g \psi^{distance}\}$.

Since we need some 'numbers' to represent island size and inter-island distance, we used the following values for each, respectively:

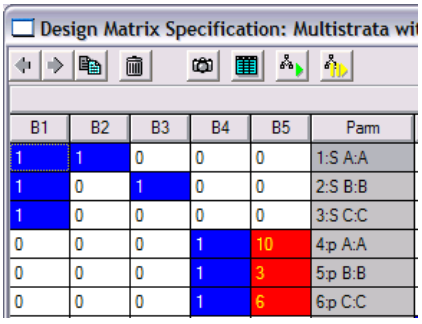
island	*size*
A	10
B	3
C	6

island	**A**	**B**	**C**
A	0	5	12
B		0	7
C			0

Since models $\{S^g p^{size} \psi^g\}$ and $\{S^g p^g \psi^{distance}\}$ are both structurally similar to model $\{S^{quality} p^g \psi^g\}$, you should be able to quickly see how to modify the design matrix (basically, as we just did, but for different parameters).

For example, consider model $\{S^g p^{size} \psi^g\}$, where we want to constrain the probability of recapture to be a function of the size of the island (where it might be reasonable to assume that the bigger the island, the lower the recapture probability for a given marked individual, all other things being equal). To fit this model, you simply need to modify the part of the design matrix corresponding to the recapture probabilities.

Given the 'island size data' in the preceding table, here is the design matrix for model $\{S^g p^{size} \psi^g\}$ (only that part of the design matrix corresponding to the survival and recapture parameters is shown).



Pretty straightforward. Potentially the only tricky one is model $\{S^g p^g \psi^{distance}\}$. Again, the key is to look closely at the coding for the movement parameters, ψ. If you're constraining ψ to be a function

of the distance, then you would replace the '1' and '0' dummy variables in the design matrix with the actual distance values themselves (remember: the '0' and '1' coding treated islands as levels of a classification factor, while using the distances directly is considering them as linear covariates). Sounds reasonable - however, this is a good example of a problem that is in fact a bit trickier than it might seem at first. For example, you need to decide if you want the probability of moving from **A** to **C** (ψ^{AC}) to be the same as the probability of moving from **C** to **A** (ψ^{CA}), since clearly the distance is the same between the same two islands, regardless of the direction you're moving. Is this a reasonable constraint?

Let's assume we want to allow the movement probabilities to differ, even among 'complementary' transitions (i.e., we'll let ψ^{AC} differ from ψ^{CA}). How would we set that up? Well, the most flexible way would be to categorize each of the movement transitions according to the donor island. For example, treating movements from island **A** as one group, from island **B** as one group, and so on. Since there are 3 island groups, then 1 intercept column, and 2 columns of dummy variables to code for island. Then, a single covariate column coding for the linear distance among islands. Finally, 2 columns for the interaction of 'island group' with linear distance. The relevant portions of the design matrix we need for this model is shown at the top of the next page. The values of the covariates are the inter-island distance values listed in the table at the top of this page:

7:Psi A to B	1	1	1	5	5	5
8:Psi A to C	1	1	1	12	12	12
9:Psi B to A	1	1	0	5	5	0
10:Psi B to C	1	1	0	7	7	0
11:Psi C to A	1	0	1	12	0	12
12:Psi C to B	1	0	1	7	0	7

Now, it is important here to understand what we've done in the design matrix. The intercept is in the first column, the dummy variables for 'island grouping' are in columns two and three, and the linear covariate (distances among islands) is in column four. The interaction of 'island' and 'distance' is shown in columns five and six. Remember, the interaction means that the estimate of movement rate varies as a unique function of island and distance among islands. Go ahead and run the model corresponding to this design matrix, and add the results to the browser. See if you can build all the models listed in the table at the bottom of p. 16.

begin sidebar

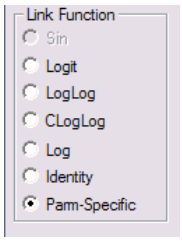
The multinomial logit link and MS models

Logically, the transitions from a given state must logically sum to 1.0. However, for reasons related to how **MARK** numerically calculates the estimates of the various parameters, this logical constraint isn't always met - in other words, the sum is occasionally > 1, which is clearly not logically feasible. This tends to happen (if it happens at all) if some of the transitions are close to the $[0, 1]$ boundary.

One solution is to change the link function **MARK** uses - from the normal sin or logit link, to what is known as the multinomial logit link function (`MLogit`). We will introduce the MLogit link here with respect to multi-state models, but it is also frequently used in POPAN (J-S) models (Chapter 13), and open robust design models (Chapter 15). The multinomial logit works as follows. Assume that each of the transition parameters from state **A** have their own β value, so that **A** to **B** is β_1, **A** to **C** is β_2, and **A** to **D** is β_3. To constrain these 3 parameters to sum to ≤ 1, the multinomial logit link works as follows:

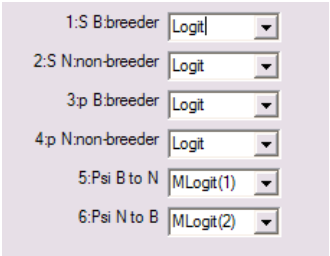
$$\psi^{AB} = \frac{e^{\beta_1}}{1 + e^{\beta_1} + e^{\beta_2} + e^{\beta_3}} \qquad \psi^{AC} = \frac{e^{\beta_2}}{1 + e^{\beta_1} + e^{\beta_2} + e^{\beta_3}} \qquad \psi^{AD} = \frac{e^{\beta_3}}{1 + e^{\beta_1} + e^{\beta_2} + e^{\beta_3}}$$

To create this set of links, you need to tell **MARK** to use a Mlogit link. You do this by first selecting the '**Parm-Specific**' link function from list of link options on the '**Run**' window:



When you hit the '**OK to run**' button, you're presented with a second window, which allows you to specify the link function for each parameter in your model (this window was first described in Chapter 6 when we introduced the cumulative logit link).

For example, in the deer example, for model $\{S^g p^g \psi^g\}$ you have 6 parameters, so the relevant part of the window looks like:

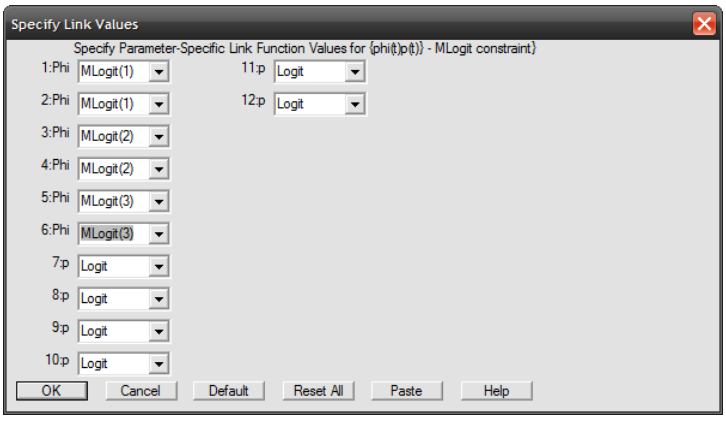


Here, we've selected **MLogit(1)** for '**Psi B to N**', and **MLogit(2)** for '**Psi N to B**'. Basically, the number inside the parentheses is a simple indexing for state (2 states - index 1 and index 2). So, if we had 3 states (**A**, **B** and **C**), we'd need 3 levels of indexing. Thus, for example, if we use **MLogit(1)** for all of the **A** state transitions, we might use **MLogit(2)** for state **B** transitions, and **MLogit(3)** for state **C** transitions.

In summary, for each set of parameters where you want the constraint that the parameters sum to ≤ 1, you must specify a **MLogit(x)** function, where '**x**' represents the set number of the MLogit link function.

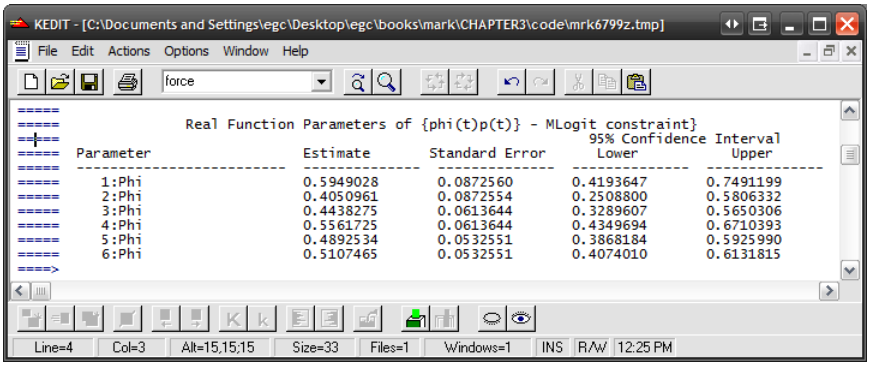
Still not clear? Here is a simple demonstration, unrelated to MS models, but using a very familiar example - the male Dipper data. Recall that there are 7 sample occasions for the Dipper data - so 6 intervals. Suppose for some reason you wanted the sum of the estimates $\varphi_1 + \varphi_2 = 1$, $\varphi_3 + \varphi_4 = 1$, and $\varphi_5 + \varphi_6 = 1$.

All you need to do to enforce these constraints using the MLogit is start with a model where apparent survival (φ) is time-dependent, then specify the '**Parm-Specific**' option from the '**Setup Numerical Estimation Run**' window. Since there are 3 sets of parameters we want to constraint (i.e., φ_1 and φ_2, φ_3 and φ_4 and φ_5 and φ_6), then we use indexing $1 \rightarrow 3$ when we specify the MLogit link for each successive pair of parameters (**MLogit(1)** for φ_1 and φ_2, **MLogit(2)** for φ_3 and φ_4, and **MLogit(3)** for φ_5 and φ_6), shown at the top of the next page:

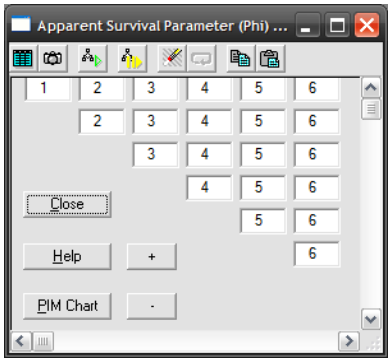


Looking at the reconstituted estimates on the real probability scale (top of the next page), we see that $\hat{\varphi}_1 + \hat{\varphi}_2 = 0.5949 + 0.4051 = 1.0000$, $\hat{\varphi}_3 + \hat{\varphi}_4 = 0.4438 + 0.5562 = 1.0000$, and $\hat{\varphi}_5 + \hat{\varphi}_6 = 0.4893 + 0.5107 = 1.0000$, as expected.

While the preceding example was obviously contrived, it serves to demonstrate the basic idea, and mechanics, behind the MLogit link function.

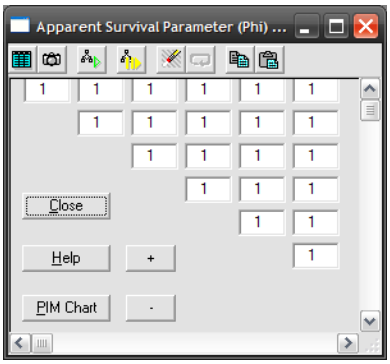


However, while specifying the MLogit link in **MARK** is straightforward, you need to be somewhat careful. Consider the following set of parameters in a PIM for the apparent survival probability for the male Dipper data:



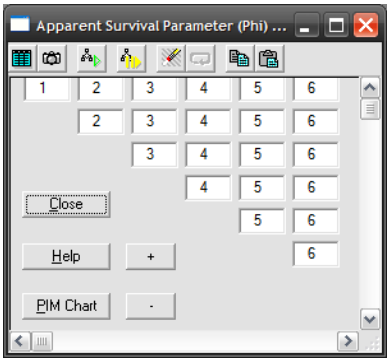
The parameter-specific link would be selected in the '**`Setup Numerical Estimation Run`**' window, and the **`MLogit(1)`** link would be applied to parameters 1 → 6 to force these 6 estimates to sum to ≤ 1.

But suppose that instead you wanted to force *all* of the 6 survival probabilities to be the same, and have the sum of all 6 be ≤ 1? You might be tempted to specify a PIM such as



(i.e., simply use the same index value for all the parameters in the PIM, which would result in the same estimate for each interval), and apply the MLogit link to parameter 1, but that would be incorrect. Changing the PIM and selecting the MLogit link for parameter 1 would result in parameter 1 alone summing to ≤ 1 (i.e., just like a logit link), but would *not* force the sum of the 6 values of parameter 1 to sum to ≤ 1. Go ahead and try it for yourself - you'll see that whether or not you use the MLogit or logit link, parameter 1 is estimated (for a model with time-varying encounter probability) as 0.5561. Clearly, $(6 \times 0.5561) \gg 1$.

But, what if you want the sum of the 6 estimates to be 1.0? To implement such a model, the PIM should not be changed from the time-dependent PIM shown on the previous page (i.e., it should maintain the indexing from $1 \rightarrow 6$); instead, the *design matrix* should be used to force the same estimate for parameters $1 \rightarrow 6$:



Then the **`MLogit(1)`** link should be specified for all 6 parameters for apparent survival, $1 \rightarrow 6$. The result is that now all 6 parameters have the same value ($\hat{\varphi} = 0.16\dot{6}$ for the model specified in this DM - with time-dependent encounter probability), where $(6 \times 0.16\dot{6}) = 1$.

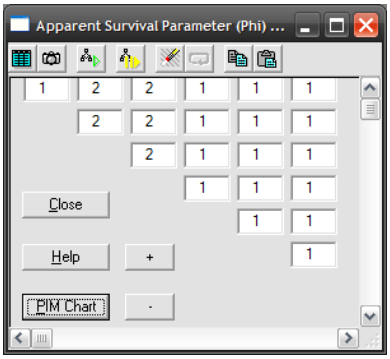
Another example - suppose you wanted parameters 1 and $4 \rightarrow 6$ (non-flood years) to be the same value, and parameters 2 and 3 (flood years) to be the same value. Obviously, there are multiple ways to implement such a model in **MARK** - the approach you use will be determined by what constraints you want to implement. If you want to have a separate estimate of apparent survival for flood and non-flood years, and (i) have the same estimate for all flood and non-flood years, and (ii) have the estimates within a flood-type sum to 1.0, then you need to use the design matrix, and apply the MLogit link function to the appropriate parameters. If we apply **`MLogit(1)`** to flood years (parameters $1, 4 \rightarrow 6$), and **`MLogit(2)`**

to non-flood years (parameters 2 and 3), we end up with estimates of apparent survival of 0.25 for each of the 4 flood years ($4 \times 0.25 = 1.0$), and 0.4965 for each of the 2 flood years ($2 \times 0.4965 = 0.9930 \leq 1.0$).

What happens if we apply a single MLogit constraint to all 6 parameters (i.e., **`MLogit(1)`** to parameters $1 \rightarrow 6$)? As you might expect, applying this constraint will still yield separate estimates of apparent survival for all flood and non-flood years, but now, the sum of estimates over all 6 parameters will be ≤ 1. What we see if we run this model, with **`MLogit(1)`** applied to parameters $1 \rightarrow 6$ is $\hat{\varphi}_{flood} = 0.1857$, and $\hat{\varphi}_{non-flood} = 0.1286$. Summing over all estimates, $(4 \times 0.1857) + (2 \times 0.1286) = 1.0$.

Now, final test - what if you want to have a single separate estimate for flood and non-flood years, and have them sum to 1.0? In other words, instead of generating an estimate for each year subject to the constraint, you want to generate an estimate for each flood type subject to the constraint (so, 2 estimates, not 6, with the sum of the estimates ≤ 1). With a bit of thought, you should realize that to fit this particular model, you do, in fact, need to first modify the PIM - which ultimately controls the number of estimated parameters - and then apply the Mlogit constraint to the 2 parameters specified in the PIM.

Here is the modified PIM - parameter 1 indicating the flood years, and parameter 2 indicating the non-flood years:



If we run this model without applying the MLogit constraint, using instead the standard logit link, we see that we obtain estimates of $\hat{\varphi}_{flood} = 0.5970$, and $\hat{\varphi}_{non-flood} = 0.4725$. Note that the sum of these estimates $0.5970 + 0.4725 = 1.07 > 1.0$. Now, if we re-run the analysis, but apply the MLogit constraint to both parameters (i.e., `MLogit(1)` for both parameters 1 and 2), we obtain estimates of $\hat{\varphi}_{flood} = 0.5728$, and $\hat{\varphi}_{non-flood} = 0.4272$ - the sum of these constrained estimates is $(0.5728 + 0.4272) = 1.0$, as expected.

The key point with these examples is that the PIM *cannot* be used to constrain parameters if you want the entire set of parameters (i.e., over all intervals) to sum to ≤ 1. Rather, the design matrix has to be used to make the constraints, with each of the entries in the PIM given the same **`MLogit(x)`** link.

end sidebar

10.4. Multi-state models as a unifying framework

Multi-state models offer great potential to increase our understanding of complex, structured systems - systems with multiple states, and stochastic (or probabilistic) transitions among states. **MARK** makes it fairly straightforward to fit some relatively complex models to the underlying multi-state structure.

This point was noted in a paper by Lebreton, Almeras & Pradel (1999)*, who pointed out that multi-state modeling does, in fact, have the potential to be a common, unified 'framework' under which a large

* Lebreton, Almeras & Pradel (1999) Competing events, mixtures of information and multi-stratum recapture models. *Bird Study*, **46** (supplment), S39-S46.

variety of models can be fit - including those combining information from multiple sources. Using data from multiple types will be discussed in a later chapter - for the moment, we'll introduce the conceptual framework described by Lebreton *et al*., to give you the sense of 'how it is done', and (with a bit of thought) how easy it is to implement.

10.4.1. Simple example (1) – CJS mark-recapture as a MS problem

We'll start by considering a simple mark-recapture analysis, based on live-encounter data. While we already have plenty of tools to handle these sorts of data, the simplicity of this data type, and your familiarity with it, make it a good starting point. First, keep in mind that the multi-state approach considers multiple states. In a mark-recapture analysis (or any simple survival analysis), there are 2 states of interest: live, and dead. As noted by Lebreton *et al*, the interesting 'paradoxical' issue with mark-recapture analysis is that the information on survival (or, equivalently, mortality) is not based on observations of dead animals. Some animals are in fact in the 'dead' state, but are never seen. So, if a 'dead' state animal is never seen, then clearly, the recapture probability for this state is 0. This leads quite logically to a fairly straightforward representation of the CJS model as a 2-state model (`Alive=1, Dead=2`), with a 0 probability of capture in the second state (i.e., when dead, or state 2, p=0).

As such, we can define the following transition probabilities. Let φ_i = probability of surviving from time i to i+1. Thus, the probability of surviving and moving from state 1 to 1 (i.e., from live → live) is clearly φ. The probability of moving from state 1 to 2 (live → dead) is $(1 - \varphi)$. The probability of moving from dead to live is clearly 0. And the probability of moving from dead to dead is 1 (i.e., if you're already dead, then you will be dead at the next occasion also). Now, if you're in state 1 (live), the recapture probability is p, while if you're in state 2, the recapture probability is 0.

We can express these transitions in a 2-state transition matrix $\mathbf{\Psi}$, and the recapture probabilities in a 2-state vector, $\mathbf{p}$:

$$\mathbf{p} = \begin{matrix} 1 & 2 \end{matrix} \begin{bmatrix} p & 0 \end{bmatrix} \qquad \mathbf{\Psi} = \begin{matrix} \\ 1 \\ 2 \end{matrix} \begin{matrix} \begin{matrix} 1 & 2 \end{matrix} \\ \begin{bmatrix} \varphi & 0 \\ 1-\varphi & 1 \end{bmatrix} \end{matrix}$$

For the transition matrix $\mathbf{\Psi}$, the rows correspond to the state at time i+1 (live and dead for rows 1 and 2), and the columns correspond to the state at time i (live and dead for columns 1 and 2). Note that the matrix must be constrained to have the sum of each column be equal to 1. Using '`Alive`' = state 1, and '`Dead`' = state 2, a typical capture history might be '`00110100`'. No '`2`' ever appears since that state is never observed.

To demonstrate this analysis, we simulated a basic CJS data set (`CJS_MS.INP`) - 8 occasions, big sample sizes, with the following parameter values:

	occasion						
	1	2	3	4	5	6	7
φ	0.5	0.85	0.85	0.65	0.50	0.60	0.85
p	0.45	0.45	0.55	0.75	0.75	0.45	0.55

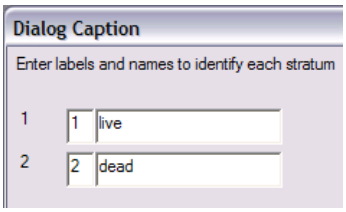
So, basic time dependence in both survival and recapture probability - model $\{\varphi_t p_t\}$. Estimates from fitting this model to the data using a 'normal' live-encounter CJS approach are shown at the top of the next page.

CJS version

Real Function Parameters of {phi(t)p(t)}

Parameter	Estimate	Standard Error	95% Confidence Interval Lower	Upper
1:Phi	0.4924350	0.0134292	0.4661590	0.5187528
2:Phi	0.8373917	0.0123089	0.8118035	0.8601003
3:Phi	0.8574654	0.0116202	0.8331397	0.8787609
4:Phi	0.6528416	0.0096733	0.6336476	0.6715512
5:Phi	0.5018881	0.0087432	0.4847559	0.5190158
6:Phi	0.6045285	0.0126119	0.5795616	0.6289620
7:Phi	0.6824350	57.774503	0.2553029E-226	1.0000000
8:p	0.4198697	0.0164394	0.3880324	0.4523881
9:p	0.4447301	0.0106304	0.4240035	0.4656508
10:p	0.5606805	0.0095834	0.5418188	0.5793685
11:p	0.7510784	0.0097206	0.7315430	0.7696398
12:p	0.7608846	0.0109886	0.7386873	0.7817535
13:p	0.4555198	0.0114734	0.4331374	0.4780832
14:p	0.6821416	57.749647	0.4130439E-226	1.0000000

Now let's analyze these data using a multi-state approach. Start **MARK**, and select the '**multi-state data type**'. Specify 8 occasions, and 2 states - the `.INP` file uses classic '`0`' or '`1`' coding for a recapture data set, so change state **A** to 1, and label it 'live', and state **B** to 2, and label it 'dead':

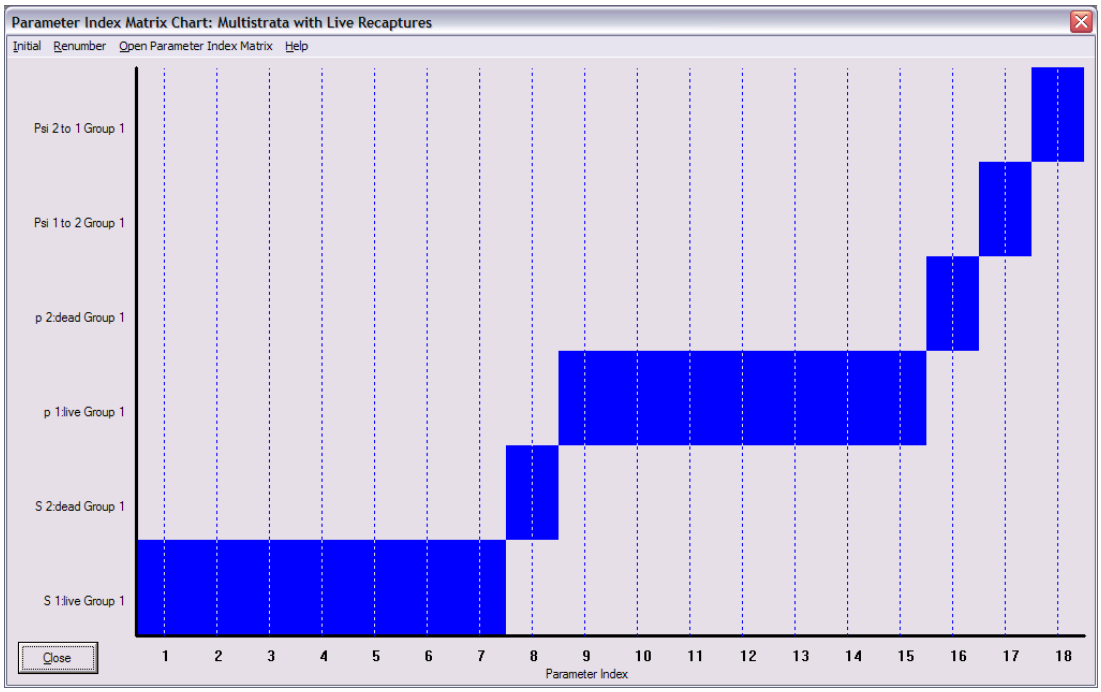


Now, the only potentially 'tricky' part of the analysis. Open up the PIM chart. You'll see that there are the 6 blue boxes - 2 for survival, 2 for recapture, and 2 for movement, for the 2 states, respectively. Now, some thinking. Based on the preceding page, we know that recapture probability for dead individuals (state 2) is 0 (in other words, we assume that we never see dead individuals-in effect, we're modeling movement into an 'unobservable state'). So, right click the blue box corresponding to recapture probability for that state, and set it 'constant' - we will fix the parameter value later, during the numerical estimation run. Renumber it with (or without) overlap - doesn't much matter in this case.

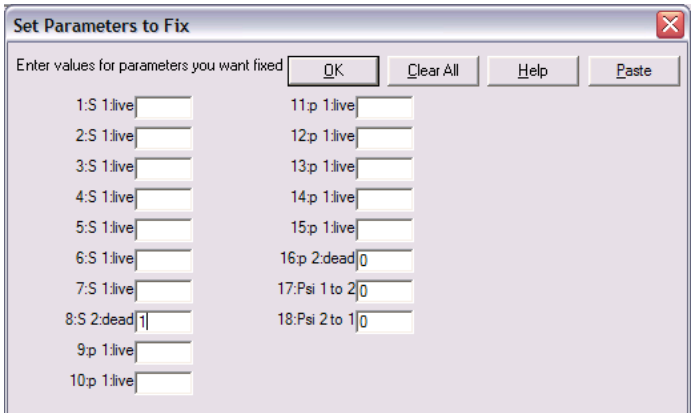
Next, we know that the probability of moving from dead to live is 0. So, we set this transition (from state 2 to state 1) constant - again, we will fix the value of this parameter to be 0 during the numerical estimation run. We also need to set the survival of state 2 individuals (i.e., dead individuals) constant - once dead, always dead, so the probability of surviving and staying in this state is clearly 1. Now, while this might seem a bit strange at first, remember that we are considering probabilities of survival and movement between states. Go ahead and modify the PIM chart, followed by renumbering to eliminate the 'spaces' between the blue boxes (alternatively, you can manually drag the boxes so they are effectively contiguous).

Now, for the last step – and one you might have to think about for a minute. What about the movement parameter from state 1 to 2 (i.e, live to dead)? Clearly this movement (from live to dead) is a logical complement to the probability of dying, which is defined as $(1 - \varphi)$. So, in order for an individual to enter state 2, it must die. State 2 is clearly an 'absorbing state' - once entered, it can't be left. Thus, the probability of moving from state 1 to state 2, conditional on surviving from i to i+1 (which is the assumption we're making throughout) is clearly 0 in this case. If you survive from i to i+1, then you

clearly can't move from state 1 (live) to state 2 (dead)! Read this section again - a couple of times if needed - to make sure you have the logic down. Once you've gotten a good grip on the idea, simply modify the PIM chart accordingly - set the parameter structure for the movement probability from state 1 (live) to state 2 (dead) to be constant. The PIM chart is shown below:

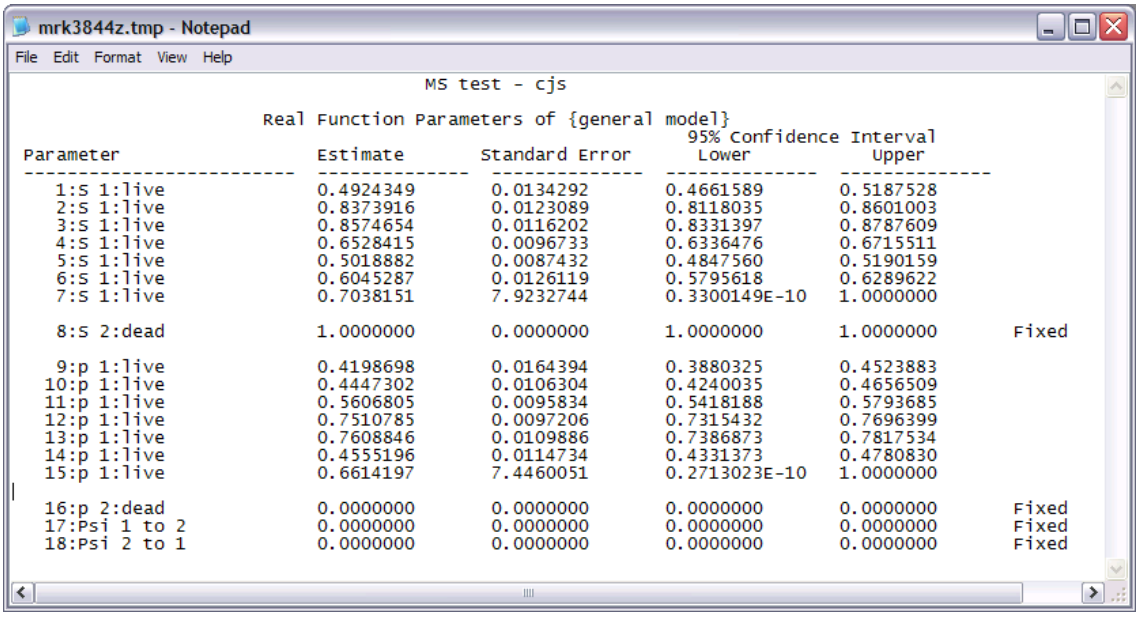


That's about it. Now, the only thing we need to do is run the model, after fixing some of the parameters. Which ones? From the PIM chart, we want to set parameter 8 (the 'survival' probability of the individuals in the dead state) to be 1, and parameters 16, 17, and 18 (the encounter probability for dead individuals, and the movement probabilities - which are conditional on survival) to be 0. To do this, run the model, and click the '**`Fix Parameters`**' button. This will bring up a small dialog window which will ask you to enter the value to which you want certain parameters to be fixed:



Note that we fix a parameter to a certain value simply by entering that value in the appropriate space.

Go ahead, run the model, and look at the reconstituted estimates:



mrk3844z.tmp - Notepad

File Edit Format View Help

MS test - cjs

Real Function Parameters of {general model}

Parameter	Estimate	Standard Error	95% Confidence Interval Lower	Upper	
1:S 1:live	0.4924349	0.0134292	0.4661589	0.5187528	
2:S 1:live	0.8373916	0.0123089	0.8118035	0.8601003	
3:S 1:live	0.8574654	0.0116202	0.8331397	0.8787609	
4:S 1:live	0.6528415	0.0096733	0.6336476	0.6715511	
5:S 1:live	0.5018882	0.0087432	0.4847560	0.5190159	
6:S 1:live	0.6045287	0.0126119	0.5795618	0.6289622	
7:S 1:live	0.7038151	7.9232744	0.3300149E-10	1.0000000	
8:S 2:dead	1.0000000	0.0000000	1.0000000	1.0000000	Fixed
9:p 1:live	0.4198698	0.0164394	0.3880325	0.4523883	
10:p 1:live	0.4447302	0.0106304	0.4240035	0.4656509	
11:p 1:live	0.5606805	0.0095834	0.5418188	0.5793685	
12:p 1:live	0.7510785	0.0097206	0.7315432	0.7696399	
13:p 1:live	0.7608846	0.0109886	0.7386873	0.7817534	
14:p 1:live	0.4555196	0.0114734	0.4331373	0.4780830	
15:p 1:live	0.6614197	7.4460051	0.2713023E-10	1.0000000	
16:p 2:dead	0.0000000	0.0000000	0.0000000	0.0000000	Fixed
17:Psi 1 to 2	0.0000000	0.0000000	0.0000000	0.0000000	Fixed
18:Psi 2 to 1	0.0000000	0.0000000	0.0000000	0.0000000	Fixed

The estimates for survival and recapture probability are identical to what we observed using a 'normal' CJS approach to this analysis (which we presented a couple of pages back), as they should be. Again, at this point, you might be asking yourself - why bother with a multi-state approach to this analysis, when we could have just as easily done it (in fact, more easily) using the standard 'recapture' analysis? The reason is - if you understand this multi-state approach in this simple case, then it won't be too difficult to see how it can be applied to other data types – for example, dead recovery data, our next example.

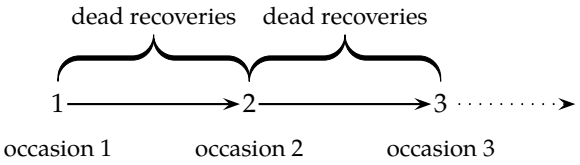
10.4.2. Simple example (2) – dead-recovery analysis as a MS problem

Lebreton, Almeras & Pradel (1999) showed that analysis of dead recoveries can be naturally reframed as multi-state capture-recapture models with two discrete states (live and dead). When framing recovery models as multi-state models, death (mortality) is represented by the transition between 'live' and 'dead', where the 'dead' state is then an 'absorbing state' (meaning, entry into the 'dead' state is permanent). The main challenges in implementing them in **MARK** are to some degree 'conceptual', and 'mechanical' (in particular, the re-formatting of the `.inp` file that will be required).

In multi-state models, animals move and are potentially detected ('encountered' in the broad sense) in 1 of N possible states. They are parameterized in terms of an $(N \times N)$ transition matrix $\mathbf{\Psi}$, an $(N \times 1)$ vector $\mathbf{p}$ of capture probabilities, and a $(N \times 1)$ vector $\mathbf{S}$ of survival probabilities. This is the ($\mathbf{S}$,$\mathbf{\Psi}$, $\mathbf{p}$) parametrization. This framework can be used for band-recovery models if one defines 2 discrete states: newly marked (say, 'banded') alive (state **B**) and newly-dead (state **N**).

However, for multi-state analysis of dead recovery data, building encounter histories takes a bit of thought. While marking and live re-encounters occur during discrete, short periods (i.e., at discrete 'encounter occasions'), dead recoveries occur throughout the interval between discrete live marking/encounter occasions. of the interval between two occasions.

Consider the following diagram – dead recoveries occur between discrete sampling occasions (1, 2, 3...).



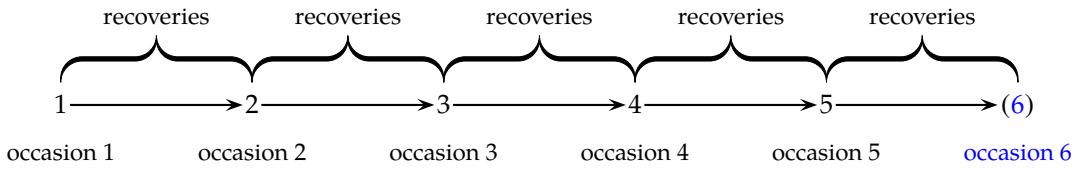
Since MS models are used to estimate transition probabilities among discrete states, then we clearly need to 'discretize' time of entry into the 'newly dead' state. To do this, individuals recovered between occasion i and $i + 1$ will be coded as entering the state 'newly-dead' (**N**) at the discrete occasion $i + 1$. For instance, with a study of 4 occasions, the capture history for an animal marked and released alive at occasion 1 and recovered dead between time 2 and 3 would then be '`A0N0`'. Note that if we were using standard `LDLD` coding (chapter 2, chapter 8), then this individual history would be coded '`10000100`'. And, in fact, this is the first challenge in using a MS approach to analyze dead recovery data in **MARK** – you first need to reformat your data such that dead recoveries are coded as entering the new state (in this case, the newly dead state **N**) at the end of the interval during which they were recovered. If you have some 'programming skills', this isn't perhaps too difficult. But, it is necessary.

To give you a sense of what is required, consider the following data set coded in standard `LDLD` format (shown at the top of the next page): the data consists of 5 occasions of mark and release, and 5 intervals during which dead recoveries can occur.

```
1000000000 1649;
1001000000 74;
1000010000 61;
1100000000 134;
1000000001 40;
1000000100 42;
0010000000 1667;
0010010000 90;
0010000001 52;
0011000000 114;
0010000100 77;
0000100000 1746;
0000110000 100;
0000100100 80;
0000100001 74;
0000001000 1770;
0000001100 108;
0000001001 122;
0000000010 1866;
0000000011 134;
```

So, how do we 'transform' these `LDLD` encounter histories to MS format? The first thing we have to remember is that individuals recovered between occasion i and $i + 1$ will be coded as entering the state 'newly-dead' (**N**) at the discrete occasion $i + 1$. Sounds straightforward, but what about the final occasion/interval?

In fact, with 5 mark and release occasions, we will need to restructure our MS encounter histories to have 6 occasions – 5 true mark and release occasion ($1 \rightarrow 5$), and one 'virtual' occasion (6) which represents the 'occasion' on which the newly dead and recovered during the interval following the occasion 5 are tabulated (as entering the newly dead state **N**):



Once you have your brain wrapped around this basic idea, the rest is relatively easy. Take the LDLD encounter history '1001000000'. It might help to 'separate' the LD pairs: '10 01 00 00 00'. So, based on the LDLD format, we see 5 such pairs. How do we transform this to the MS format? First, the initial 1 for any history will always represent the newly marked individuals - so, we simply change the initial 1 to B. So, our history is now 'B0 01 00 00 00'. Next, we consider that each 'LD' pair represents a single occasion in the MS format, and that the information in the 'LD' pair is collapsed from 2 characters to 1. So, 'B0 01 00 00 00' becomes 'B 1 0 0 0'. Next, the '1' indicating dead recovery during the second interval is changed to N, to indicating entering the newly dead state. So, 'B 1 0 0 0' becomes 'B N 0 0 0'. Finally, we add a 6th occasion to handle any dead recoveries that might occur in the interval following the 5th release occasion: 'B N 0 0 0 0'. All that is left is to eliminate the spaces in the encounter history, so that we're left with 'BN0000'.

So, here is the final transformed input file (make sure you see the connection between each of these transformed encounter histories and the corresponding history in LDLD format shown above).

```
B00000 1649;
B0D000 74;
B00D00 61;
BD0000 134;
B0000D 40;
B000D0 42;
0B0000 1667;
0B0D00 90;
0B000D 52;
0BD000 114;
0B00D0 77;
00B000 1746;
00BD00 100;
00B0D0 80;
00B00D 74;
000B00 1770;
000BD0 108;
000B0D 122;
0000B0 1866;
0000BD 134;
```

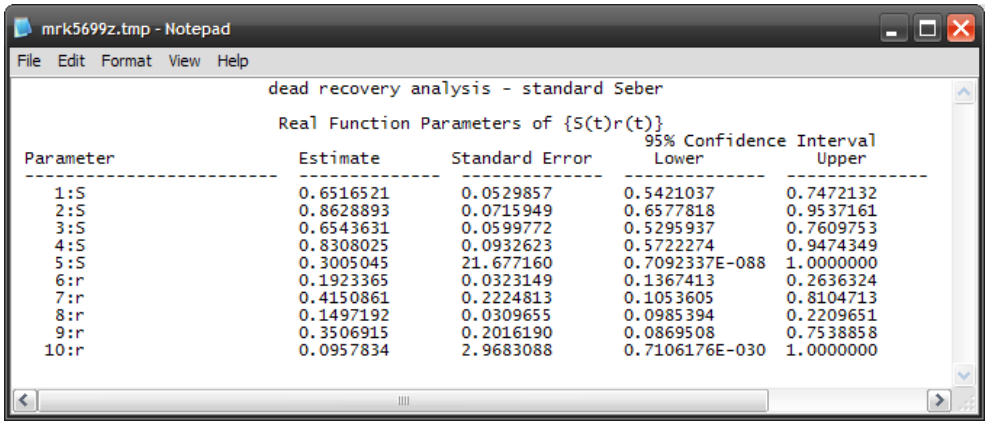
OK – now that we have our 'transformed' data file ready, what next? Well, now we move from a 'mechanical' consideration (data formatting) to a 'conceptual' one.

Here is the 2-state transition matrix $\mathbf{\Psi}$, and the 2-state survival and encounter vectors, $\mathbf{S}$ and $\mathbf{p}$:

$$\mathbf{\Psi} = \begin{array}{c} \\ \mathbf{B} \\ \mathbf{N} \end{array} \begin{array}{c} \begin{array}{cc} \mathbf{B} & \mathbf{N} \end{array} \\ \begin{bmatrix} \psi^{11} & 1-\psi^{11} \\ 0 & 1 \end{bmatrix} \end{array} \qquad \mathbf{S} = \begin{array}{c} \\ \mathbf{B} \\ \mathbf{N} \end{array} \begin{bmatrix} 1 \\ 0 \end{bmatrix} \qquad \mathbf{p} = \begin{array}{c} \begin{array}{cc} \mathbf{B} & \mathbf{N} \end{array} \\ \begin{bmatrix} 0 & r \end{bmatrix} \end{array}$$

By fixing the first entry of the $\mathbf{S}$ vector to 1, then the elements of the first row of $\mathbf{\Psi}$ represent survival S (element ψ^{11}) and mortality $1 - S$ (element $1 - \psi^{11}$), respectively. In the second row of the matrix, the transition from state $\mathbf{N} \rightarrow \mathbf{B}$ is logically fixed to 0, and thus its complement (from $\mathbf{N} \rightarrow \mathbf{N}$) is by default 1. Now, a subtle point. In theory, a third state (which we might call 'permanently dead') is needed because an animal can be recovered only once in the $\mathbf{N}$ state (i.e., the year it dies). Failing to consider such a 'permanently dead' state would mean that an individual entering the $\mathbf{N}$ state could be recovered forever, which is clearly a logical impossibility. However, because such a 'permanently dead' state to which individuals move immediately after death contributes no information (i.e., it is never encountered and transitions to this state are all fixed), it can be ignored. We simply need to fix the survival of the state $\mathbf{N}$ to 0 in the $\mathbf{S}$ matrix.

Now, we're ready to proceed with the analysis. First, for purposes of comparison, we'll start with a 'classical' dead recovery analysis of these data, using the Seber parameterization. We'll fit the default time-dependent model $\{S_t r_t\}$ to these data. The model deviance was 5.8316. Parameter estimates are shown below:



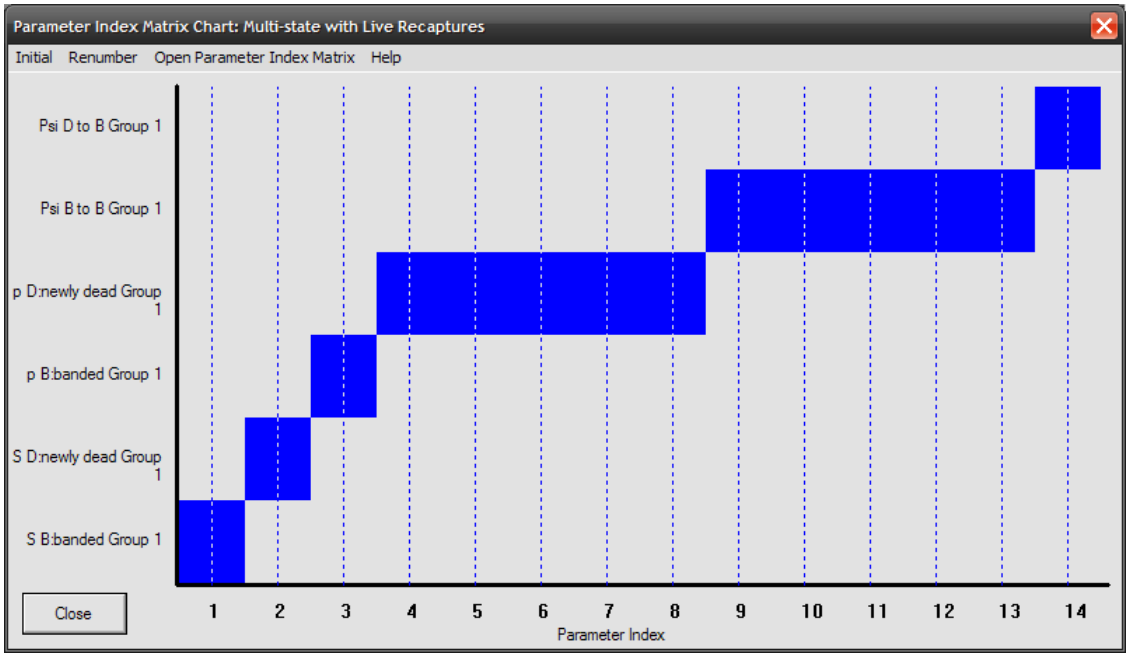
As expected for a fully time-dependent model, the final estimates of S and r are confounded.

Now, let's fit these data using a MS approach. We have to start a new project in **MARK**, specifying a multi-state data type with 2 states (**B** and **N**), and 6 occasions. Since we want to estimate survival (and not mortality), change the default PIM definition so the ψ^{BD} is estimated by subtraction.

Next, we need to modify the PIM structure to represent the transition structure for the 3 parameters in our model. Recall that the model we're trying to fit is $\{S_t r_t\}$.

$$\mathbf{\Psi} = \begin{array}{c} \\ \mathbf{B} \\ \mathbf{N} \end{array} \begin{array}{c} \begin{array}{cc} \mathbf{B} & \mathbf{N} \end{array} \\ \begin{bmatrix} \psi^{11} & 1-\psi^{11} \\ 0 & 1 \end{bmatrix} \end{array} \qquad \mathbf{S} = \begin{array}{c} \\ \mathbf{B} \\ \mathbf{N} \end{array} \begin{bmatrix} 1 \\ 0 \end{bmatrix} \qquad \mathbf{p} = \begin{array}{c} \begin{array}{cc} \mathbf{B} & \mathbf{N} \end{array} \\ \begin{bmatrix} 0 & r \end{bmatrix} \end{array}$$

The multi-state PIM structure for this model (given these transitions) is



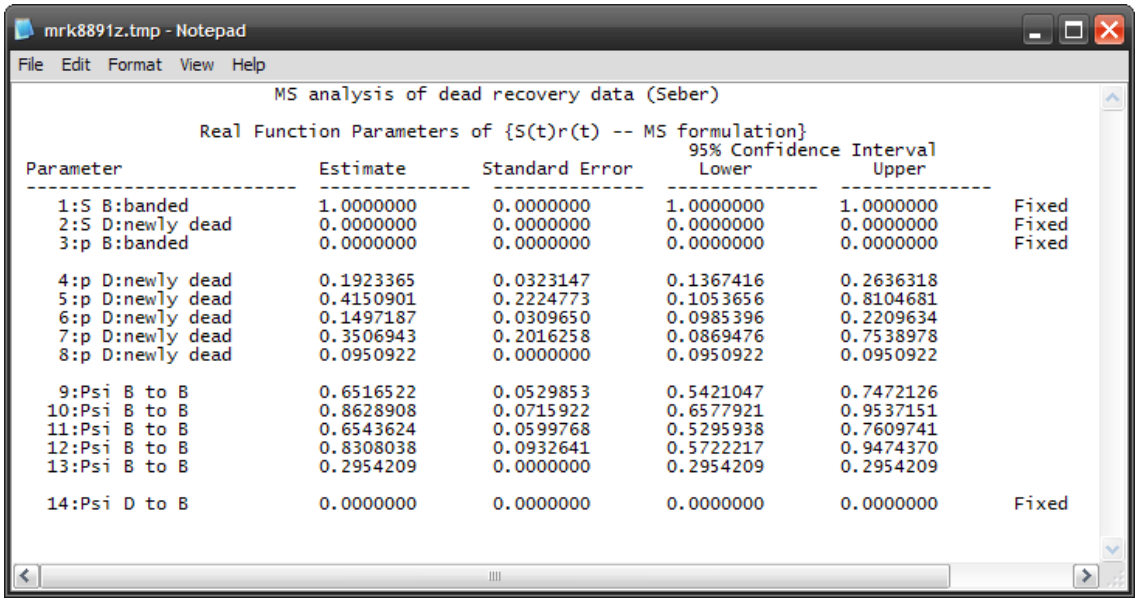
As a check, notice that the PIM structure has only 2 time-dependent parameters (ψ^{BB}, representing survival, and p^{D}, represent the recovery probability r) – all the other parameters are fixed constants. This makes sense, since the model we're trying to fit $\{S_t r_t\}$ clearly has only two parameters, both time-dependent.

Before submitting the model, we need to fix some parameters. From the transition structures on the preceding page, we see that we fix survival for the newly banded individuals in state **B** (i.e., parameter 1) to 1, and the survival for newly dead individual in state **N** (i.e., parameter 2) to 0. Clearly, the encounter probability for the newly banded individuals (i.e., parameter 3) is fixed to 0. Finally, the probability of moving from state **D** to **B** (i.e., from dead to live; parameter 14) is fixed to 0. After fixing the parameters, we run the model. Model deviance is reported as 5.8316, which is identical to the deviance reported for the same model using the classical dead recovery analysis. The parameter estimates from the MS analysis are shown at the top of the next page. If you compare these estimates from those from the classical dead recovery analysis (top of the preceding page), you'll see they are identical (at least for the non-confounded parameters). Pretty slick, eh? We'll leave it to you to figure out how to extend this to a joint live encounter-dead recovery analysis (see Lebreton *et al*. 1999 and Lebreton & Pradel 2002* for details).

10.4.3. A more complex example - recruitment probability

To really make the flexibility of this approach clear, we'll now look at an example related to the earlier question concerning different breeding states (breeding, and non-breeding), but with a twist - here we're going to look at recruitment, which we'll define as the probability of moving between a non-breeder to a first time breeder. Now, unlike the analysis of breeding state we presented earlier in the chapter, but

* Lebreton, J-D., Pradel, R. (2001) Multi-state recapture models: modelling incomplete individual histories. *Journal of Applied Statistics*, **29**, 353-369.



mrk8891z.tmp - Notepad

File Edit Format View Help

MS analysis of dead recovery data (Seber)

Real Function Parameters of {S(t)r(t) -- MS formulation}

Parameter	Estimate	Standard Error	95% Confidence Interval Lower	Upper	
1:S B:banded	1.0000000	0.0000000	1.0000000	1.0000000	Fixed
2:S D:newly dead	0.0000000	0.0000000	0.0000000	0.0000000	Fixed
3:p B:banded	0.0000000	0.0000000	0.0000000	0.0000000	Fixed
4:p D:newly dead	0.1923365	0.0323147	0.1367416	0.2636318	
5:p D:newly dead	0.4150901	0.2224773	0.1053656	0.8104681	
6:p D:newly dead	0.1497187	0.0309650	0.0985396	0.2209634	
7:p D:newly dead	0.3506943	0.2016258	0.0869476	0.7538978	
8:p D:newly dead	0.0950922	0.0000000	0.0950922	0.0950922	
9:Psi B to B	0.6516522	0.0529853	0.5421047	0.7472126	
10:Psi B to B	0.8628908	0.0715922	0.6577921	0.9537151	
11:Psi B to B	0.6543624	0.0599768	0.5295938	0.7609741	
12:Psi B to B	0.8308038	0.0932641	0.5722217	0.9474370	
13:Psi B to B	0.2954209	0.0000000	0.2954209	0.2954209	
14:Psi D to B	0.0000000	0.0000000	0.0000000	0.0000000	Fixed

analogous to the multi-state approach to recapture analysis we just completed, we're interested here in a 'permanent' state transition. In the recapture analysis, we were interested in the transition from 'live' to 'dead'. The various constraints we imposed during the numerical estimation reflect the fact that the transition is permanent (once dead, always dead). Here in this example, we're also considering a permanent state transition - from 'non-breeder', defined as a bird which has *never* bred, to a 'breeder', or (perhaps more accurately, a 'recruit') - an individual which has become a breeder (i.e., has bred at least once). Now, once an individual is a breeder, it is always a breeder. It may not breed in every year following first breeding, but it is always a breeder.

This question, and various approaches to estimation of the probability of making this transition from non-breeder to recruit, have been discussed at length by Pradel & Lebreton (*Bird Study* **46**: S74-81), and references therein. Our purpose here is merely to point out again how the multi-state approach can be used to deal with situations where there is a non-observable state. What is the non-observable state? In this case, it is the non-breeding (pre-recruitment) state (which we'll call **NB**). In many species, only breeding individuals are encountered. Thus, we need to separate the effects of being a **NB** individual (for which $p = 0$) from death. The multi-state approach is one way to tackle this problem.

Consider the following situation (as described by Pradel and Lebreton). The probability of making the permanent transition from non-breeding to breeding (i.e., the probability of recruiting) is governed by the parameter a. Formally, let a_i be the probability that an as yet inexperienced individual of age i starts to breed at that age. An individual is marked as a newborn, and then each year, you go out to look for that individual. If you encounter the individual, then it has both survived, and recruited. If you don't encounter the individual, it is either because it hasn't survived, or that it hasn't recruited (and is thus non-observable). Take the encounter history '2011', where '2' denotes the birth date (time of initial marking), and '1' denotes 'breeding' or 'recruited'. Assume there are only 2 age-specific survival probabilities (φ_j and φ_a), a constant recapture probability p, and age-specific probabilities of recruitment (a_1, a_2 and a_3). Thus, the associated probability of this encounter history is $\varphi_j[(1-a_1)a_2 + a_1(1-p)]\varphi_a p_. \varphi_a p_.$.

Pradel and Lebreton then simulated a data set, setting $\varphi_j = 0.4$, $\varphi_a = 0.8$, $p = 0.5$. They assumed that survival did not differ among pre-recruits (non-breeders) and recruits. For age-specific recruitment probabilities, they used $a_1 = 0.2$, $a_2 = 0.375$, and $a_3 = 1.0$.

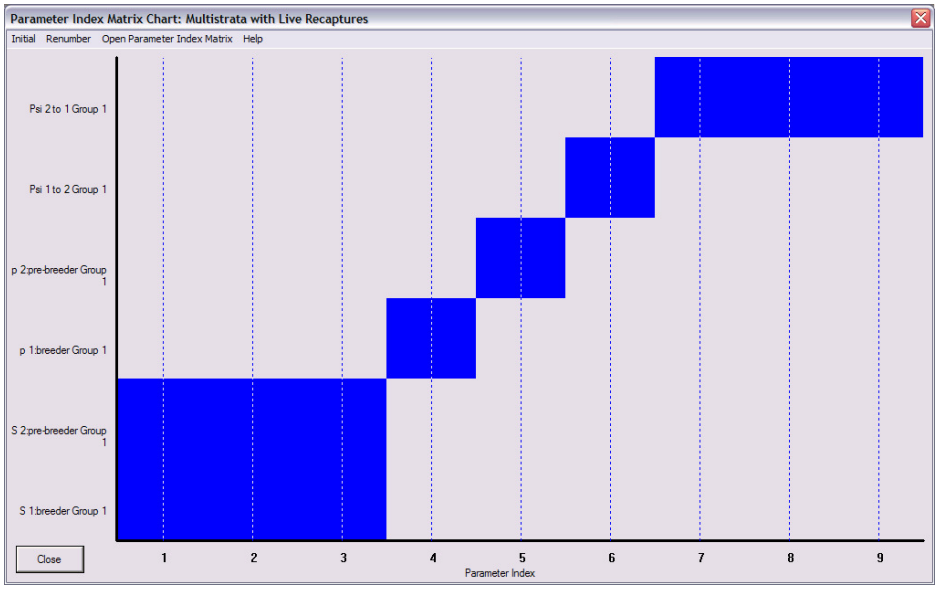
Simulating the encounter histories for a single cohort of 5,000 individuals, they arrived at the following histories:

history	*frequency*	*history*	*frequency*
2111	32	2011	128
2110	48	2010	192
2101	32	2001	448
2100	88	2000	4032

Let's analyze these data using **MARK**. The transition matrices governing these data are:

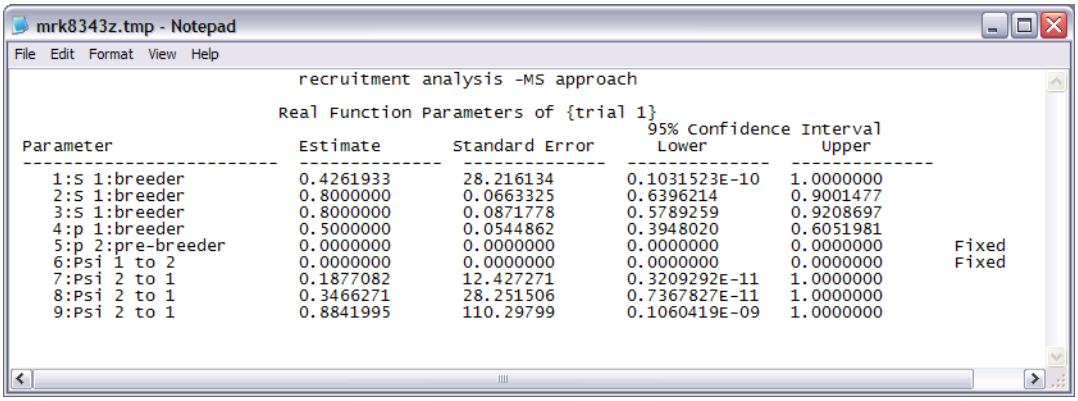
$$\begin{bmatrix} 1-a_i & 0 \\ a_i & 1 \end{bmatrix} \begin{bmatrix} \varphi_i \\ \varphi_i \end{bmatrix} \begin{bmatrix} 0 \\ p_i \end{bmatrix}$$

The procedure for analyzing these data using **MARK** is mechanically similar to what we just did when using the multi-state approach to analyze live recapture data. The only difference here is that we have to impose different constraints. The intellectual challenge here is thinking about what constraints to impose, and whether or not they make 'biological' sense (as opposed to constraints that are needed out of structural necessity to make one or more parameters identifiable). We 'know' from the simulated data that recapture probability is constant, so we'll modify the PIM chart to reflect this. We also know that the probability of capturing a non-breeding pre-recruit is 0, so we set recapture probability for non-breeders to be a constant (we'll fix it to 0 just before the numerical estimation run). We assume that survival for breeders (recruits) and non-breeders (pre-recruits) is age-specific and the same for both groups (remember that within a single cohort, age and time are equivalent). We also know the probability of moving from breeder to non-breeder is 0, so we will set that parameter to be 0 before the numerical estimation. Finally, the '*a*' parameter we're really interested in – this corresponds to the probability of moving from non-breeder to breeder. We 'believe' that this is likely to be age-specific, so we use a simple time-specific parameterizations for this probability (remember, 'age = time within cohort', and here, we're dealing with a single cohort). Here is the PIM chart:



We proceed to run the numerical estimation, first fixing the recapture probability for non-breeders (parameter 5 in the PIM chart) to 0, and the probability of moving from breeder back to non-breeder (parameter 6 in the PIM chart) to 0.

So far, so good - but what about the probability of moving from non-breeder to breeder (parameter a)? It might seem *a priori* there is no need to fix any of this parameters - after all, these are the parameters we're interested in estimating in the first place. However, we see quickly that fixing only parameters 5 and 6 as described leads to all sorts of problems.


mrk8343z.tmp - Notepad

File Edit Format View Help

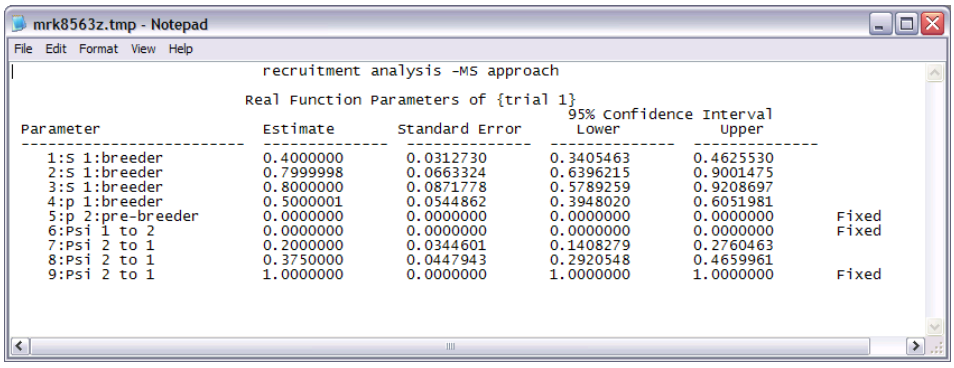
recruitment analysis -MS approach

Real Function Parameters of {trial 1}

Parameter	Estimate	Standard Error	95% Confidence Interval Lower	Upper	
1:S 1:breeder	0.4261933	28.216134	0.1031523E-10	1.0000000	
2:S 1:breeder	0.8000000	0.0663325	0.6396214	0.9001477	
3:S 1:breeder	0.8000000	0.0871778	0.5789259	0.9208697	
4:p 1:breeder	0.5000000	0.0544862	0.3948020	0.6051981	
5:p 2:pre-breeder	0.0000000	0.0000000	0.0000000	0.0000000	Fixed
6:Psi 1 to 2	0.0000000	0.0000000	0.0000000	0.0000000	Fixed
7:Psi 2 to 1	0.1877082	12.427271	0.3209292E-11	1.0000000	
8:Psi 2 to 1	0.3466271	28.251506	0.7367827E-11	1.0000000	
9:Psi 2 to 1	0.8841995	110.29799	0.1060419E-09	1.0000000	

Although numerical convergence is reached, the estimates for virtually all of the parameters are clearly wrong - anything estimated with a standard error of 0 or a 95% CI from $0 \rightarrow 1$ is clearly wrong (even with simulated data!). Nonetheless, note that the estimates themselves 'seem' to match the 'true' values used in the simulation rather well, except for φ_j, and the final recruitment probability a_3. This sort of thing often implies that one (or more) parameters are not identifiable under the given set of constraints - either because the constraints are incorrect, or because you haven't constrained enough parameters. In this case, it turns out that the latter is the cause of the problems here.*

What other constraint do we need? Well, as it turns out, we need to impose a logical constraint that by age 3, all individuals in the population that are still alive will have recruited. This is explained in detail in the Pradel & Lebreton paper. The results from this 'logically constrained' analysis are shown below:


mrk8563z.tmp - Notepad

File Edit Format View Help

recruitment analysis -MS approach

Real Function Parameters of {trial 1}

Parameter	Estimate	Standard Error	95% Confidence Interval Lower	Upper	
1:S 1:breeder	0.4000000	0.0312730	0.3405463	0.4625530	
2:S 1:breeder	0.7999998	0.0663324	0.6396215	0.9001475	
3:S 1:breeder	0.8000000	0.0871778	0.5789259	0.9208697	
4:p 1:breeder	0.5000001	0.0544862	0.3948020	0.6051981	
5:p 2:pre-breeder	0.0000000	0.0000000	0.0000000	0.0000000	Fixed
6:Psi 1 to 2	0.0000000	0.0000000	0.0000000	0.0000000	Fixed
7:Psi 2 to 1	0.2000000	0.0344601	0.1408279	0.2760463	
8:Psi 2 to 1	0.3750000	0.0447943	0.2920548	0.4659961	
9:Psi 2 to 1	1.0000000	0.0000000	1.0000000	1.0000000	Fixed

Note, in fact, that the data were simulated assuming $a_3 = 1.0$ in the first place. If we re-run the analysis, setting parameter 5 and parameter 6 to 0, and also setting parameter 9 to 1, we see that our

* The identification of parameters that are not estimable given the structure of the model is discussed in Appendix F.

estimates are now 'correct'.

Now, clearly, the key step was assuming that there was an age after which all individuals were recruited, conditional on still being alive. This is a strong assumption, and one that makes application of this approach somewhat problematic - see Pradel & Lebreton for a full discussion. However, the point here is to demonstrate the methodology, and not to provide a full treatment of this complex problem. As noted by Pradel & Lebreton, the multi-state approach may have utility for this particular analysis, given some assumptions. But, more importantly, we again see how useful the multi-state approach can be in dealing with states which are not observable.

begin sidebar

Estimation problems: local minima – approaches and solutions

In the preceding example, our estimates were 'wonky' (nonsensical) until we applied the appropriate and necessary logical constraints to the model. However, this is not always the case. Sometimes, multi-state models have 'problems' converging to global maxima. In fact, the multi-state model can display some heinous behavior when there are > 2 states, most notably multiple optima in the likelihood function (Lebreton & Pradel 2002). That is, depending on the starting values used to maximize the likelihood function, the solution can vary. Most of the models we have explored so far have a single maximum, and so this behavior is not encountered.

There are several approaches to handling these sort of problems, which we describe here. First, for multi-state models in particular, it is not a bad idea to first build a simple, time-constant 'dot' model containing all the factors of interest. We will use this model to generate 'good starting values' for the the more general model. Note that previously, we advocated first building the most general model in your candidate model set, for which you will then estimate $\hat{c}$ (see chapter 5). In general, this works fine. But for multi-state models in particular, and other data types where you might be having some 'numerical estimation problems' for your general model, using starting values from a simpler model often helps you avoid some problems.

In some cases, though, even very simple models will have problems. A second approach then is to make use of a different link function. In some cases, a different link function may do a 'better job' at navigating what might be a multi-modal likelihood surface than another. In fact, if you find fitting a given model to a data set using different link functions yields very different model deviances, then this is a reasonable indicator that you might be having problems finding the global minima with some link functions.

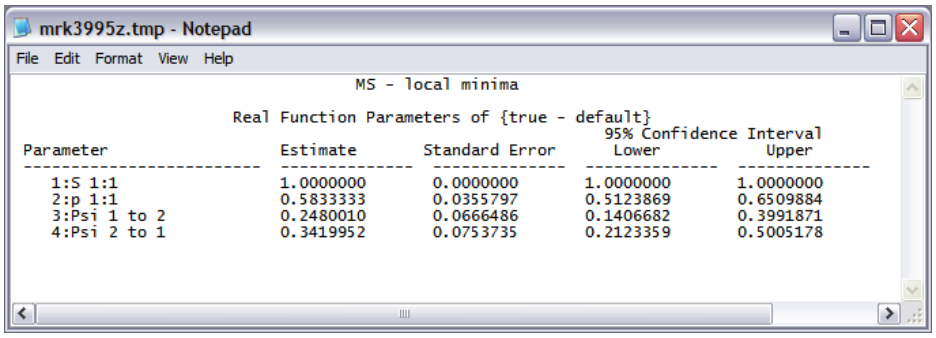
Finally, you might try using an alternative optimization procedure available in **MARK**. This procedure, based on *simulated annealing*, is selected by clicking the '**alternate optimization**' check-box on the right-hand side of the '**run numerical estimation**' window. Simulated annealing (Goffe *et al*. 1994) is less computationally efficient than the default optimization algorithm in finding the maximum of the function, typically requiring many more evaluations of the likelihood to reach a solution. However, the reason for this 'inefficiency' is why simulated annealing is provided in **MARK**; periodically, the simulated annealing algorithm makes a random jump to a new parameter value, and this characteristic is what allows the algorithm more flexibility in finding the global maximum (instead of getting stuck at a local maximum).

We will demonstrate the various approaches to handling 'estimation problems' using an example multi-state data set with a known local minimum: 2 states (1 and 2), 7 occasions (this example was extracted from an example from Jerome Dupuis, used in a recent paper by Gimenez *et al*. 2005 - **JABES**). Here are the encounter histories:

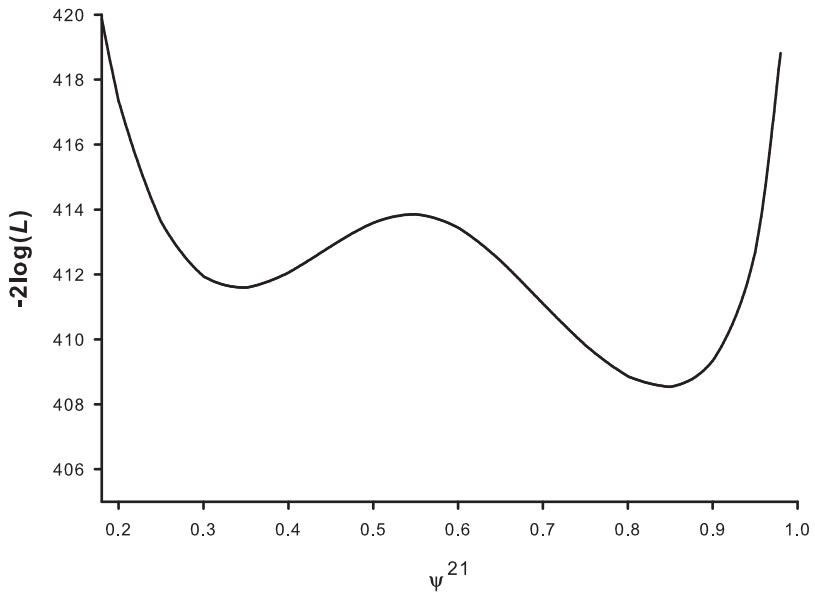
```
2021202 4;          1110101 4;
2020201 4;          1010101 4;
2020202 4;          1010102 4;
2201021 4;          2102011 4;
```

The true parameter values are $S = 1.0$ (constant over time, and the same for both states), $p = 0.6$

(constant over time, and the same for both states), $\psi^{12} = 0.60$ (constant over time), and $\psi^{21} = 0.85$ (constant over time). If you fit model $\{S\ p\ .\ \psi_s\}$ (which is the true model) to the data in **MARK**, using the default numerical optimization routine, we get the following estimates:

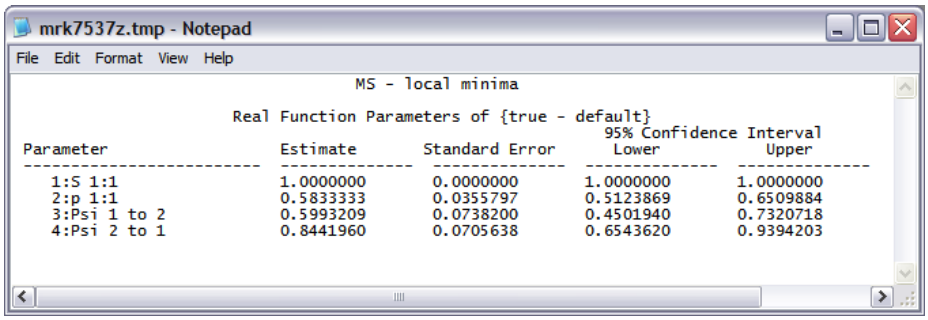


We see clearly that **MARK** has had some problems (although we know this only because we know what the true values are). What has happened? Well, let's have a look at the likelihood profile for values of ψ^{21} from $0.2 \rightarrow 1.0$.



We see from the likelihood profile that there are in fact two local minima: one at approximately 0.35, and another at approximately 0.85. Now, look back at our estimate for ψ^{21} generated using the default numerical optimization routine - we see that $\hat{\psi}^{21} = 0.342$, which is roughly where the first local (non-global) minima occurs. In this case, it was a case of 'bad luck' that **MARK** converged to this local minima - the bad luck owing to the starting values **MARK** defaults to.

If we use a starting value of 0.85 for ψ^{21}, and try again, we see that now **MARK** 'correctly' gives us the correct parameter estimates:


mrk7537z.tmp - Notepad

File Edit Format View Help

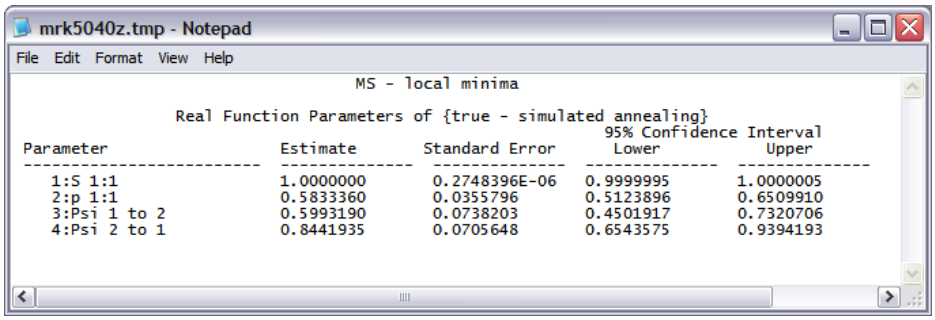
MS - local minima

Real Function Parameters of {true - default}

Parameter	Estimate	Standard Error	95% Confidence Interval Lower	Upper
1:S 1:1	1.0000000	0.0000000	1.0000000	1.0000000
2:p 1:1	0.5833333	0.0355797	0.5123869	0.6509884
3:Psi 1 to 2	0.5993209	0.0738200	0.4501940	0.7320718
4:Psi 2 to 1	0.8441960	0.0705638	0.6543620	0.9394203

Of course, in practice, we won't be able to 'cheat' by using starting values close to the true values - since we won't know what the true parameter values are (obviously)! So, what can we do? Well, if you're willing to get your computer to do a bit of work for you, you can make use of the alternate optimization capability in **MARK** - based on simulated annealing. Recall that, periodically, the simulated annealing algorithm makes a totally random jump to a new parameter value. It is this characteristic that allows the annealing algorithm to be more likely to find the global maximum instead of a local maximum.

For our example, when we run the simulated annealing algorithm, we see that **MARK** gives us the correct estimates - even using the default starting values:


mrk5040z.tmp - Notepad

File Edit Format View Help

MS - local minima

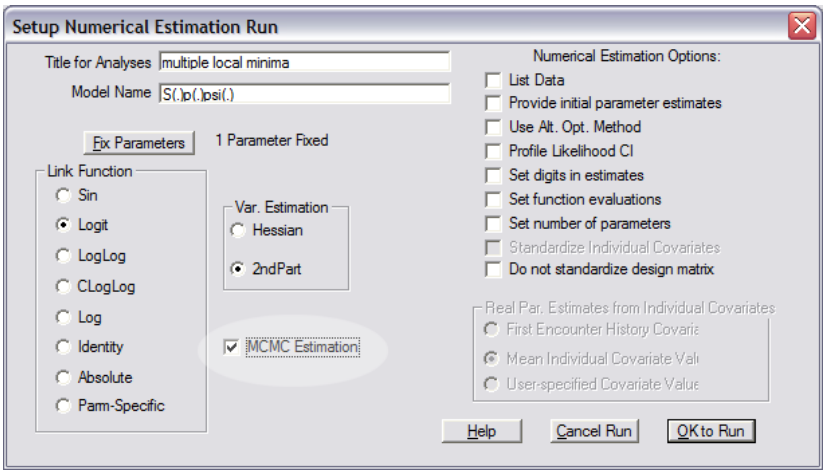
Real Function Parameters of {true - simulated annealing}

Parameter	Estimate	Standard Error	95% Confidence Interval Lower	Upper
1:S 1:1	1.0000000	0.2748396E-06	0.9999995	1.0000005
2:p 1:1	0.5833360	0.0355796	0.5123896	0.6509910
3:Psi 1 to 2	0.5993190	0.0738203	0.4501917	0.7320706
4:Psi 2 to 1	0.8441935	0.0705648	0.6543575	0.9394193

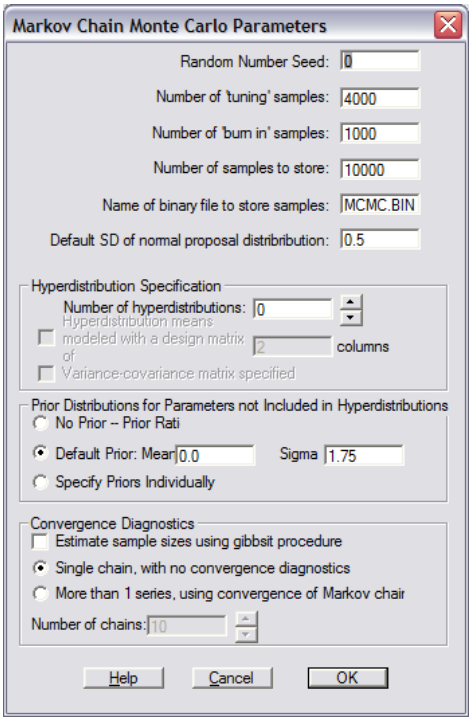
However, as noted earlier, simulated annealing is **much** slower than the standard numerical optimization routine **MARK** uses. So, be warned. But, if you have a strong suspicion that one or more of your parameter estimates are at the boundary, then there is some chance you might have a local minima (or several, especially if you have > 2 states), and simulated annealing might be your best option.

How can you 'predict' when you might have such local minima? Unfortunately, there is no known simple diagnostic you can apply *a priori* (although there does seem to be some evidence that such local minima are more likely to occur for time-dependent models with > 2 states). However, by making using of a numerically intensive approach known as *Markov chain Monte Carlo* (MCMC), we can evaluate the *posterior distribution* to determine whether or not there may be local minima in the likelihood for a given parameter (if the notion of MCMC, and posterior distributions, are new to you, no need to worry - these are covered in detail in Appendix E). Here, we will simply demonstrate the mechanics of using MCMC in **MARK**, using the preceding example where we have already determined there to be local minima in the likelihood.

To use MCMC estimation in **MARK** you first need to check the '**MCMC Estimation**' check-box in the '**numerical estimation run**' window:



Then, once you've clicked the '**OK to run**' button, a window will be spawned asking you to specify the MCMC parameters (shown below).

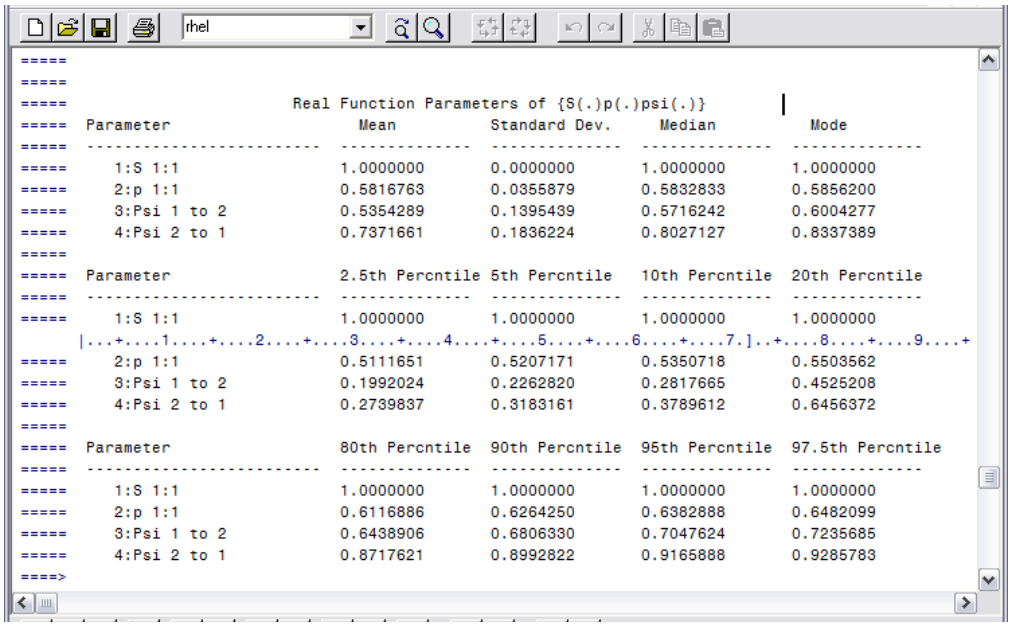


For the moment, we'll simply accept the defaults (the details of the various MCMC parameters will be covered in the pending appendix) - these are *usually* sufficient to give us 'a reasonable' look at the posterior, from which we can often determine the presence of local minima in the likelihood for a given parameter. Note that the default file where the MCMC samples will be stored is the file `MCMC.BIN` (we need these samples to construct the posterior). The `MCMC.BIN` file is created in whatever directory contains, the `.INP`, `.DBF`, and `.FPT` files associated with the data set you're working with.

Once you click the '**OK**' button, **MARK** will spawn a numerical estimation window - you can 'watch' as **MARK** iterates through the MCMC samples (where the number of iterations is equal to the number of burn in samples (1000, by default), plus the number of 'tuning' samples (4000, by default), plus the number of post-burn and post-tuning samples (10000, by default) - so, accepting the default parameters, 15,000 total samples (iterations of the Markov chain). Once **MARK** has finished, it will spawn a window showing various summary statistics calculated from the posterior distribution for each of your parameters. These summary statistics are presented for both the parameter estimates on the transformed scale (*note*: for MCMC estimation in **MARK**, the default settings for the priors assume you're using the logit link. If you change to another link function, you'll probably want to modify the priors as well), followed by the same summary statistics for the parameters transformed back to the real probability scale. The summary statistics for the 4 parameters for our example problem are shown at the top of the next page.

Of particular interest are the mean, median and modes of the posterior distributions, and the 2.5th and 97.5th percentiles of the posterior (from which we derive the 95% 'credibility interval' for our various parameters). Look closely at the mean, mode, and median statistics for the various parameters. Start with the encounter parameter p (we ignore survival S, here, since it was fixed to 1.0). Note that for p, the mean, median and mode are all very close to each other.

However, this is obviously not the case for parameters ψ^{12} and ψ^{21}. For ψ^{12}, for example, the mean is 0.535, the median is 0.572, and the mode is 0.600. Similarly, for ψ^{21}, the mean is 0.737, the median is 0.803, and the mode is 0.834. Recall that the 'true' parameter values were $\psi^{12} = 0.60$, and $\psi^{21} = 0.85$. Clearly, both the mean and median are not particularly robust estimators of either parameter - in both cases, the mode is the better statistic.*

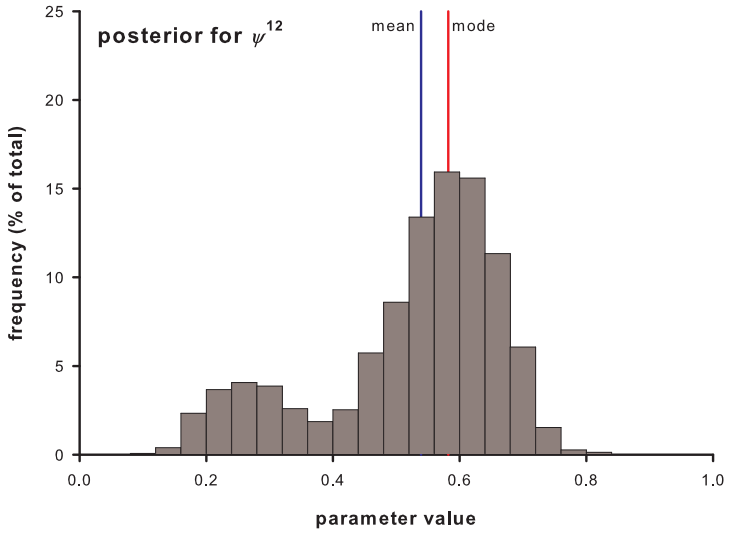

rhel

```
=====
=====
=====                    Real Function Parameters of {S(.)p(.)psi(.)}
=====  Parameter                 Mean          Standard Dev.     Median          Mode
=====  ------------------------  ------------  ------------  ------------  ------------
=====     1:S 1:1                1.0000000     0.0000000     1.0000000     1.0000000
=====     2:p 1:1                0.5816763     0.0355879     0.5832833     0.5856200
=====     3:Psi 1 to 2           0.5354289     0.1395439     0.5716242     0.6004277
=====     4:Psi 2 to 1           0.7371661     0.1836224     0.8027127     0.8337389
=====
=====  Parameter                 2.5th Percntile 5th Percntile  10th Percntile  20th Percntile
=====  ------------------------  --------------  ------------  --------------  --------------
=====     1:S 1:1                1.0000000     1.0000000     1.0000000     1.0000000
      |...+....1....+....2....+....3....+....4....+....5....+....6....+....7.]..+....8....+....9....+
=====     2:p 1:1                0.5111651     0.5207171     0.5350718     0.5503562
=====     3:Psi 1 to 2           0.1992024     0.2262820     0.2817665     0.4525208
=====     4:Psi 2 to 1           0.2739837     0.3183161     0.3789612     0.6456372
=====
=====  Parameter                 80th Percntile  90th Percntile  95th Percntile  97.5th Percntile
=====  ------------------------  --------------  --------------  --------------  --------------
=====     1:S 1:1                1.0000000     1.0000000     1.0000000     1.0000000
=====     2:p 1:1                0.6116886     0.6264250     0.6382888     0.6482099
=====     3:Psi 1 to 2           0.6438906     0.6806330     0.7047624     0.7235685
=====     4:Psi 2 to 1           0.8717621     0.8992822     0.9165888     0.9285783
====>
```

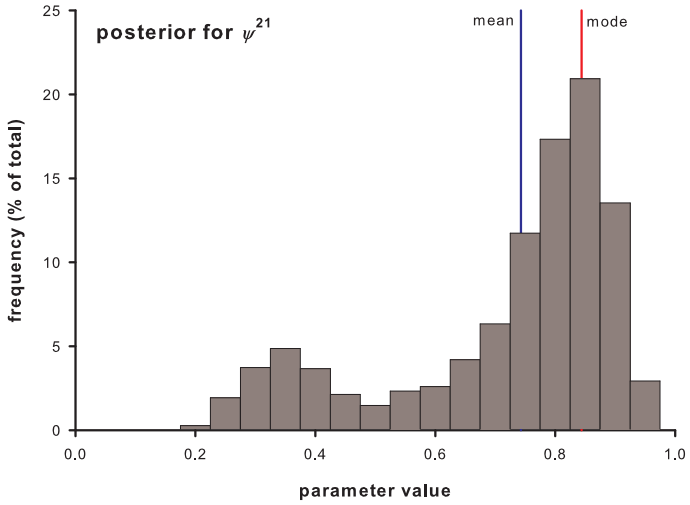
What do these differences among mean, median and mode suggest? Well, simplistically, they 'hint' at the possibility there may be 'problems' with the likelihood - specifically, multiple local minima. We can confirm our suspicion by considering two different 'plots' derived from the MCMC samples. First, consider a frequency histogram plotting the frequency (expressed as a percentage of the total) with which a particular parameter value was 'visited' during the iteration of the Markov chain.

* But we know this only because here we know the true parameter values. In fact, what we've just demonstrated is that MCMC is not necessarily a solution to in deriving point parameter estimates.

Here is the frequency histogram for the posterior chain for ψ^{12} (*note*: for a variety of reasons, the chain was 'thinned' by extracting every tenth value, prior to deriving the frequency histograms. This is commonly done to minimize the effects of serial autocorrelation in the Markov chains, although there is some debate as to whether or not this is really necessary).



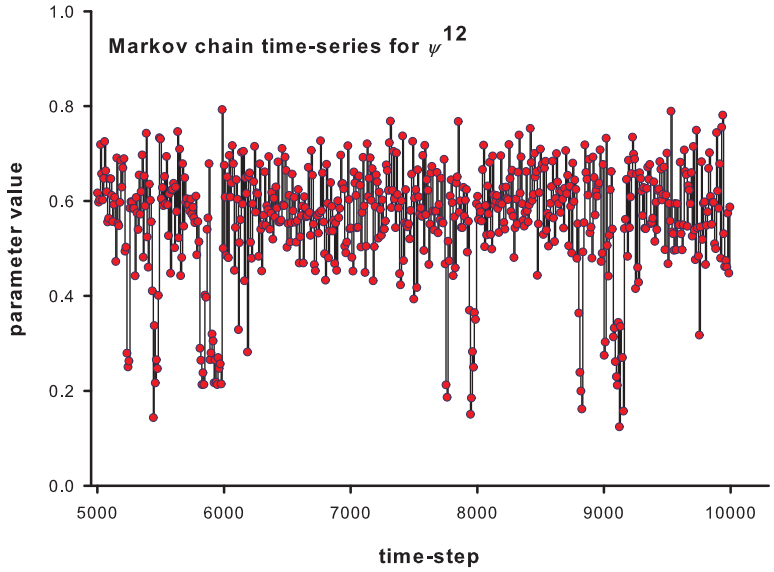
Here is the equivalent histogram for ψ^{21}:



We see clearly that the posterior distributions for both ψ^{12} and ψ^{21} are bimodal. These plots help us understand why **MARK** converged on the local minimum. For example, note that the smaller, left-hand modal point for ψ^{21} occurs at approximately 0.34-0.35, which is roughly what **MARK** reported (incorrectly) as the MLE for this parameter. The default starting value used by **MARK** (0.5) was closer to this local minimum than the global minimum for ψ^{21} (which occurs at 0.85). The same explanation holds for ψ^{12} as well. So, multi-modal frequency distributions of 'visits by the Markov chain' to parts

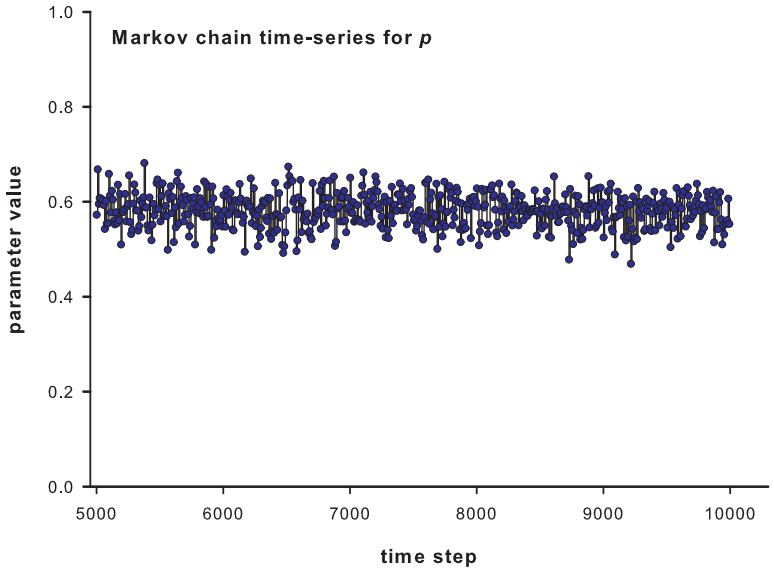
of the parameter space indicates local minima, which **MARK** may (or may not) converge to (depending on the relative proximity of the starting value to a local minima).

We can also find evidence for local minima by examining the time-series of iterations of the Markov chain. Here is a plot of part of the time-series (first 5000 iterations after burn-in and tuning) for ψ^{12}:



We see clearly that the chain periodically 'jumps' to a region of the parameter space between 0.1 and 0.4, corresponding to the smaller, left-hand mode shown on the frequency histogram on the preceding page for ψ^{12}.

Contrast this Markov chain with that generated for parameter p



For p, we see that the Markov chain is very 'tight' - indicating that the parameter will be estimated with good precision. In fact, the profile likelihood CI for p, estimated using the simulated annealing approach to estimating the likelihood, shows the 95% CI for p to be [0.513, −0.652], which is remarkably close to the MCMC-based 95% 'credibility interval', [0.511, −0.648].

From the preceding, we can draw several conclusions:

1. > 1 minima in the likelihood for a given parameter are possible for multi-state models, especially for fully time-dependent models where there are numerous states. This can cause problems, if **MARK** converges on one of these local minima (thus yielding the 'wrong' estimate for that parameter). **MARK** will give you **no** warning when this occurs.
2. you can often test for the possibility of local minima by examining the posterior distribution for a given parameter, generated using the Markov Chain Monte Carlo (MCMC) option in **MARK**. Doing so for the various parameters in your general model may be a good first step prior to fitting other models. But, be warned - MCMC estimation on all of the parameters of a 'large' general model (i.e., time-dependence, many states) can take a *long* time. Moreover, there are a number of options in using the MCMC estimation routines in **MARK** that you may have to 'tweak' in the process (these will be described in the pending appendix on MCMC estimation in **MARK**), adding to the overall time it might take to fully explore the posterior distributions for various parameters.
3. an alternative approach is to either (i) try various different starting values - if they all result in convergence to the same parameter values, then you may be fortunate and not have local minima to concern yourself with. Or, alternatively, you could (ii) use the optimization routines based on simulated annealing, which are very likely to converge - *eventually* - to the true local minima. Again - we emphasize the word 'eventually', since simulated annealing can take a *long* time to converge.

end sidebar

10.5. GOF testing and multi-state models

What about GOF (goodness of fit)? By now, you should realize that one of the first steps in any analysis is assessing the fit of the most general model in your candidate model set to the data (introduced in Chapter 5). As part of this process, we make an estimate of $\hat{c}$ (a measure of the lack of fit of the model to the data), and use this $\hat{c}$ to 'adjust' the criterion we use for model selection. Here, we will describe 2 approaches to GOF testing for multi-state models.

10.5.1. Program U-CARE and GOF for time-dependent multi-state models

A few years ago, Roger Pradel and colleagues have described a method for assessing the fit of a fully time-dependent MS model to data:

Pradel, R., C. M. A. Wintrebert & O. Gimenez. (2003) A proposal for a goodness-of-fit test to the Arnason-Schwarz multi-site capture-recapture model. *Biometrics*, **59**, 43-53.

Although the specific details are somewhat complex, the rationale for the tests proposed by Pradel and colleagues are very similar to those underlying program **RELEASE**, as applied to 'normal' CJS data: the proposed tests essentially consider the fates of individuals seen before, or not, and whether (and when) they are seen again - but, in this case, conditional on the state in which they were previously observed. The GOF test proposed by Pradel *et al.* is relevant for what they refer to as the 'Arnason-Schwarz' (AS) model (the AS model is in fact what we've been discussing in this chapter). To test the

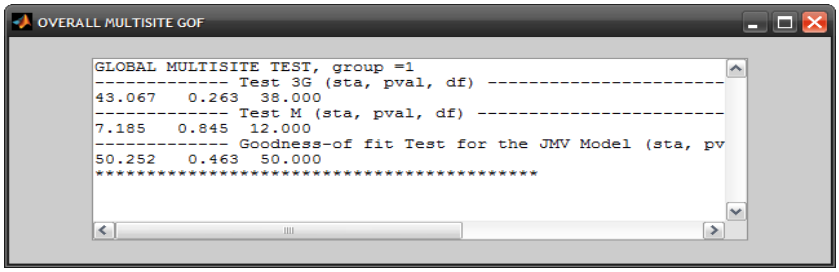
GOF of the AS model to multi-state data, Pradel *et al.* first describe a fully efficient goodness of fit test to what they refer to as a '**Jolly Move**' model (JMV). Any fully efficient GOF test is based on the property that all animals present at any given time behave in the same way. This is the basic point underlying **RELEASE**: in **Test 3**, we test the assumption of 'equivalent behaviors' whatever their past capture history, while in **Test 2**, we test the 'equivalence' assumption whether they are currently captured or not. The same logic underlies the GOF test for the JMV model.

What is this JMV model, and how does it relate to the AS model? The JMV model (Brownie *et al*. 1993) differs from the typical AS model in that it permits the capture probability for time (i+1) to depend on the state at periods (i) and (i+1) (whereas the AS model permits the encounter probability to depend only on current state, and time). Thus, the AS model is in fact a special (reduced) case of the more general JMV model. As with program **RELEASE**, a fully efficient GOF test for the JMV model is based on the property that all animals present at any given time on the same site behave the same.

Pradel *et al*. introduce 2 general tests of this multi-state 'equivalence' assumption: **Test 3G**, which assumes 'behavioral equivalence' of individuals released together regardless of their past capture history, and **Test M**, which tests 'equivalence' among those individuals that are eventually recaptured (on a subsequent occasion) conditional on whether or not they are encountered at the present occasion. (There are also 2 possible subcomponents for testing for transience, and memory models, but we will not discuss those here). See Pradel *et al*. for full details – for now, we'll focus on the basic ideas, and the mechanics

The first step in the GOF test proposed by Pradel *et al*. is to assess the fit of the fully time-dependent JMV model to the data. In many cases, the JMV model is unlikely to show significantly greater fit to the data than the AS model (since the dependence of the capture probability at time (i+1) to depend on both (i) and (i+1) is unlikely to be observed in practice very often). As such, the GOF test for the JMV model may by a generally valid test of fit for the AS model as well. Here's how you implement a GOF test to the general JMV model. First, you need a recent build of the program **U-CARE** (first described in Chapter 5). Then, you might need to modify your **MARK** input file - slightly. The only thing you need to do is make sure that all of your 'state coding' is numeric (e.g., if you use 'B' and 'N' in your input file, for example, you'll need to change them to numbers, say 1 for 'N', and 2 for 'B'...**U-CARE** cannot currently handle letters for state coding in the input file).

We'll demonstrate the use of **U-CARE** for multi-state GOF testing using simulated data (`MS_GOF.INP`). These data, consisting of 2 states, 8 occasions, were simulated under the true model $\{S_{g+t}p_{g+t}\psi_g\}$ - so additive time variation between the two states for survival, and encounter probability, but constant state differences in transition probability over time. We start **U-CARE**, and read in the input file. **U-CARE** will ask you to confirm a few things about the file (e.g., presence or absence of covariates). Once you've answered those questions, all you need to do is pull down the '**Goodness-of-Fit for Multi-state**' menu, and select '**Sum of multi-site tests**'. Selecting this option will cause **U-CARE** to fit the component tests (**3G** and **M**) and the JMV to the data. Once finished, **U-CARE** will spawn another window, giving the results of the various tests.



We see that **U-CARE** reports the 2 individual component tests (**3G** and **M**). For these simulated data, **U-CARE** reports that both tests are accepted (**Test 3G**: $\chi^2 = 33.464$, df=25, $P = 0.120$, **Test M**: $\chi^2 = 5.838$, df=10, $P = 0.829$).

The next line is the key. It reports the overall test of the JMV model to the data (basically, the sum of the 2 component tests **3G** and **M**). In this case, **U-CARE** reports that the $\chi^2 = 50.252$, with df=50. The P-value is 0.463. Thus, our estimate of $\hat{c}$ would be 50.252/50 = 1.00, which is very close to 1.0.

10.5.2. MS models and the median $\hat{c}$ test

What about the median $\hat{c}$ test introduced in Chapter 5 – can it be used for MS data? Yes, although there is limited experience with it to date. For purposes of comparison with the results from **U-CARE**, start **MARK**, and run the fully time-dependent model on these same, simulated data. Then, run the median $\hat{c}$ test - since we know these data are simulated under a reduced parameter AS model, we'll save ourselves some time and use lower bound of 1.0, and an upper bound of 2.5 for our analysis (if you don't remember the details of the median $\hat{c}$ GOF test, go back and look at Chapter 5). We'll use 5 design points, with 10 replicates at each point. Using these values, **MARK** reported an estimate of $\hat{c}$ of 0.99, which is effectively 1.0. Again, although there is little experience to date with the median $\hat{c}$ and GOF testing of MS models, preliminary results look promising. For the moment, we suggest using **U-CARE** for GOF testing, especially if your general model is the fully time-dependent model. For reduced parameter general models, it is probably worth trying the median $\hat{c}$ approach.

However, be advised that running a median-$\hat{c}$ test for multi-state models with a large number of occasions, and states, can take a **lot** of computer time (especially if you decide to use the simulated annealing algorithm - as described earlier in this chapter). For fully time-dependent models, **U-CARE** is much faster. However, if your most general model which adequately fits the data is not time-dependent, then your only real option is to run the median-$\hat{c}$ GOF test. The good news (relatively speaking) is that you need only assess GOF for the most general model in your candidate model set – so you need only to go to the trouble once.

10.6. multi-state models & unequal time intervals

Various data types in **MARK** have state transitions – clearly, the multi-state data type that is presented in this chapter, but they also arise in robust designs (Chapter 15), Barker models (Chapter 9), and multi-season occupancy models. Any data type with state transitions suffers from the same problem when the intervals between occasions are unequal (how **MARK** handles unequal intervals in general was introduced earlier in Chapter 4).

To illustrate the issue, consider the case where an encounter occasion is missing in the multi-state data type. Consider the following valid **MARK** 5-occasion multi-state encounter history '`A.A00`', where the missing occasion is shown as a 'dot' and there are 2 states, `A` and `B`, and occasions are all 1 time unit apart. To explain this 'dot', several possibilities exist, namely:

$$S_1^A \psi_1^{AA} (1 - p_2^A) S_2^A \psi_1^{AA} p_3^A \ldots \quad \text{and} \quad S_1^A \psi_1^{AB} (1 - p_2^B) S_2^B \psi_2^{BA} p_3^A \ldots$$

However, suppose that you coded the data with the 'dot' left out, and set the time intervals to 2, 1, and 1. That is, only 4 occasions are considered instead of 5. So the encounter history is now '`AA00`'. Unfortunately, this approach is going to give *very* different results from the proper parametrization above. **MARK** does not generate the probabilities for the transition to state `B` with this parametrization.

The probability of surviving from occasion 1 to occasion 2 would now be $(S_1^S)^2$, with no consideration that the animal could have moved to state B during the missing occasion. So, even the survival estimates S will be incorrect. The ψ parameters for the first interval are not comparable to the ψ parameters for the second and third intervals because they represent different time scales.

Internally, within **MARK**, the time interval correction on S remains, but all time interval corrections from ψ have been removed. The motivating logic is that when time intervals are 'ragged', e.g., 1.1, 0.9, 1.05, 0.95, it may still make sense to apply a correction to S. However, this correction is inappropriate for ψ, and may even be questionable for S. So, '**user beware**'!

10.7. Summary

That is the end of Chapter 10. We have considerably expanded the range of 'underlying' models we can fit to mark-recapture data - in particular, we've added a 'movement' parameter. We have also seen how to apply constraints to these models as easily as we did with CJS models. In fact, we've seen how we can apply movement models to other data types (live encounters, dead recoveries...), which highlights one of the singular strengths of MS models - the ability to combine sources of information in a natural and relatively intuitive framework.

CHAPTER 11

Individual covariates

In many of the analyses we've looked at so far in this book, we've partitioned variation in one or more parameters among different levels of what are commonly referred to as 'classification' factors. For example, comparing survival probabilities between male and female individuals (where 'sex' is the classification factor), good and poor breeding colonies (where 'colony' is the classification factor), among age-classes, and so on.

However, in many cases, there may be one or more factors which you might think are important determinants of variation among parameters which do not have natural 'classification' levels. For example, consider body size. It is often hypothesized that survival of individuals may be significantly influenced by individual differences in body size: smaller individuals may have different survival probabilities than larger individuals, for example. However, body size is a continuous trait, and as such, has no 'natural' classification levels. While it is possible to take individuals and classify them as 'large', 'medium' or 'small' (based on some criterion), such classifications are arbitrary. Obviously, for a continuous distribution, there are an infinite number of possible classification levels you might create. And, to some degree, your results will depend upon how many classification levels for body size (or some other continuous factor) you use, and exactly where these levels fall.

As such, it would be preferable to be able to use the real, continuous values for body size (for example) in your analysis - each individual in the data set has a particular body size, so you want to constrain the estimates of the various parameters in your model to be linear functions of one or more continuous individual covariates. The use of the word 'covariate' might tweak some memory cells – think 'analysis of covariance' (ANCOVA), which looks at the influence of one or more continuous covariates on some response variable, conditional on one or more classification variables. For example, suppose you have measured the resting pulse rate for male and female children in a given classroom. You believe that pulse rate is influenced by the sex of the individual, and their body weight. So, you might set up a linear model where `SEX` is entered as a classification variable (with 2 levels: male and female), and `WEIGHT` is entered as a continuous linear covariate. You might also include an interaction term between `SEX` and `WEIGHT`.

In analysis of data from marked individuals, you essentially do much the same thing. Of course, there are a couple of 'extra steps' in the process, but essentially, you use the same mechanics for model building and model selection we've already considered elsewhere in the book. The major differences concern: data formatting, modifying the design matrix, and reconstituting parameter estimates. We will introduce the basic ideas with a series of worked examples.

© Cooch & White (2014)

11.1. ML estimation and individual covariates

Conceptually, the idea behind modeling survival or recapture (or any other parameter) as a function of an individual covariate isn't particularly difficult. It stems from the realization that it is possible to write the likelihood as a product of individual 'contributions' to the overall likelihood. Consider the following example. Suppose you have 8 individuals, which you mark and release. You go out next year, and find 3 of them alive (we'll ignore issues of encounter probability and so forth for the moment). We know from Chapter 1 that the MLE for the estimate of survival probability S is simply $(3/8) = 0.375$. More formally, the (binomial) likelihood of observing 3 survivors out of 8 individuals marked and released is given as (where $Y = 3$, and $N = 8$):

$$\mathcal{L}(p \mid \text{data}) = \binom{N}{Y} S^Y (1-S)^{N-Y}$$

Or, dropping the binomial probability term (which is a constant, and not a function of the parameter – see Chapter 1):

$$\mathcal{L}(p \mid \text{data}) = S^Y (1-S)^{N-Y}$$

If we let $Q = (1-S)$, then we could re-write this likelihood as

$$\mathcal{L}(p \mid \text{data}) = S^Y Q^{N-Y} = S^3 Q^5$$

We could rewrite this likelihood expression as

$$\mathcal{L}(p \mid \text{data}) = S^3 Q^5 = S.S.S.Q.Q.Q.Q.Q = \prod_{i=1}^{3} S_i \prod_{i=4}^{8} Q_i$$

Alternatively, we might define a variable a, which we use to indicate whether or not the animal is found alive ($a = 1$) or dead ($a = 0$). Thus, we could write the likelihood for the ith individual as

$$\mathcal{L}(p \mid N, \{a_1, a_2, \ldots, a_8\}) = \prod_{i=1}^{8} S^{a_i} Q^{(1-a_i)}$$

Try it and confirm this is correct. Let S = the MLE = 0.375. Then, $(0.375)^3(1-0.375)^5 = 0.00503$, which is equivalent to

$$\begin{aligned}
&(0.375)^1(0.625)^{(1-1)}(0.375)^1(0.625)^{(1-1)}(0.375)^1(0.625)^{(1-1)} \\
&\times(0.375)^0(0.625)^{(1-0)}(0.375)^0(0.625)^{(1-0)}(0.375)^0(0.625)^{(1-0)}(0.375)^0(0.625)^{(1-0)}(0.375)^0(0.625)^{(1-0)} \\
&= (0.05273) \times (0.09537) \\
&= 0.00503
\end{aligned}$$

In each of these 3 forms of the likelihood the individual 'fate' has its own probability term (and the likelihood is simply the product of these individual probabilities). Written in this way there is a straightforward and perhaps somewhat obvious way to introduce individual covariates into the likelihood. All we need to do to model the survival probability of the individuals is to express the survival probability of each individual S_i as some function of an individual covariate X_i.

For example, we could use

$$S_i = \frac{e^{\beta_1+\beta_2(X_i)}}{1+e^{(\beta_1+\beta_2(X_i))}}\left(= \frac{1}{1+e^{-(\beta_1+\beta_2(X_i))}}\right)$$

with logit link function

$$\ln\left(\frac{S_i}{1-S_i}\right) = \beta_1 + \beta_2(X_i)$$

Then, we simply substitute this expression for S_i into

$$\mathcal{L}(p \mid N, \{a_1, a_2, \ldots, a_8\}) = \prod_{i=1}^{8} S^{a_i} Q^{(1-a_i)}$$

Written this way, the MLE's for the β_1 and β_2 (intercept and slope, respectively) become the focus of the estimation.

Pretty slick, eh? Well, it is, with one caveat. The likelihood expression gets 'really ugly' to write down. It becomes a very long, cumbersome expression (which fortunately **MARK** handles for us), and because of the way it is constructed, numerically deriving the estimates takes **much** longer than it does when the likelihood is not constructed from individuals. Also, there are a couple of things to keep in mind. First, it is important to realize that the survival probabilities are replaced by a logistic submodel of the individual covariate(s). Conceptually, then, every animal *i* has its own survival probability, and this may be related to the covariate. During the analysis, the covariate of the ith animal must correspond to the survival probability of that animal. **MARK** handles this, and it is this sort of 'book-keeping' that slows down the estimation (relative to analyses that don't include individual covariates).

OK – enough background. Let's look at some examples, and how you handle individual covariates in **MARK**.

11.2. Example 1 - normalizing selection on body weight

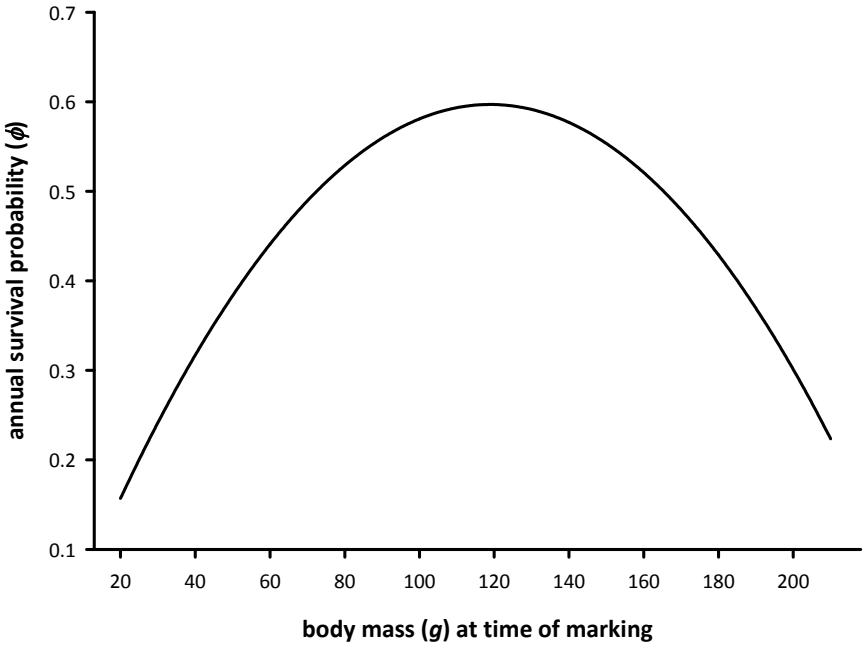
Consider the following example. You believe that the survival probability of some small bird is a function of the mass of the bird at the time it was marked. However, you believe that there might be normalizing selection on body mass, such that there is a penalty for being either 'too light' or 'too heavy', relative to some 'optimal' body mass.

Now, a key assumption - we're going to assume that survival probability for each individual bird is potentially influenced by the mass of the bird at the time it was first marked and released. Now, you might be saying to yourself 'hmmm, but body mass is likely to change from year to year?'. True - and this is an important point to keep in mind - we assume that the individual covariate (in this case, body mass) is fixed over the lifetime of the individual bird. We will consider using 'temporally variable covariates' later on. For now, we will assume that the mass of the bird when it is marked and released is the important factor.

We simulated some capture-recapture data, according to the following function relating survival probability (φ) to body mass (`mass`), according to the following equation:

$$\varphi = -0.039 + 0.0107(\texttt{mass}) - 0.000045(\texttt{mass}^2)$$

To help visualize how survival varies as a function of body mass, based on this equation, consider the following figure:



We see that survival first rises with increasing body mass, then eventually declines - this represents 'normalizing' selection, since survival is 'maximized' for birds that are neither too heavy nor too light (right about now, some of the hard core evolutionary ecologists among you may be rolling your eyes, but it is a reasonable simplification. . .).

We simulated data for 8 occasions, 500 newly marked birds per release cohort (i.e., per year). We also made our life simple (for this example) by assuming that survival probability does not vary as a function of time, only body mass. We set recapture probability to be 0.7 for all birds, whereas survival probability was set as a function of a randomly generated body mass (with mean of 110 mass units). We'll deal with the complications of time-variation in a later example.

Here is a 'piece' of the simulated data set (contained in `indcov1.inp`):

```
11111111  1  120.71  14570.24;
11111110  1   86.26   7440.76;
11111110  1  118.23  13978.42;
11111110  1   72.98   5325.47;
11111110  1  101.52  10305.69;
```

Several things to note. First, and perhaps obviously, in order to use individual covariate data, you must include the encounter history for each individual in the data file - you can't summarize your data by calculating the frequency of each encounter history as you may have done earlier (see Chapter 2 for the basic concepts if you're unsure). Each line of the `INP` file contains an individual encounter history.

The encounter history is followed immediately by a single digit '1', to indicate that the frequency of this individual history is 1 (or, that each line of data in the `.inp` file corresponds to 1 individual).

What about the next 2 columns? Consider the following line from the data file:

```
11111111  1  120.71  14570.24;
```

The values 120.71 and 14,570.24 refer to the mass of this individual bird (i.e., `mass` in the equation), and the square of the mass (i.e., `mass`^{2} in the equation - 14,570.24 = $(120.71)^2$). Now, in this example, we've 'hard-coded' the value of the square of body mass right in the `.INP` file. While this may, on occasion, be convenient, we'll see later on that there are situations where you don't want to do this, where it will be preferable to let **MARK** 'handle the calculation of the covariate functions (squaring mass, in this case) for you'.

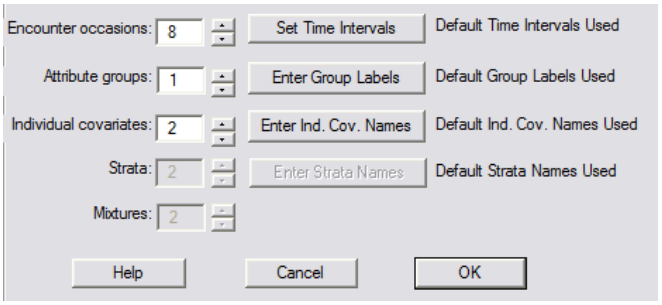
So, for each bird, we have the encounter history, the number '1' to indicate 1 bird per history, and then one or more columns of 'covariates' - these are the individual values for each bird - in this example, corresponding to mass and the square of the mass, respectively.

Finally, what about missing values? Suppose you have individual covariate data for some, but not all of the individuals in your data set. Well, unfortunately, there is no simple way to handle missing values. You can either (i) use the mean value of the covariate, calculated among all the other individuals in the data set, in place of the missing value, or (ii) discard the individual from the data set. Or, alternatively, you can discretize the covariates, and use a multi-state approach. The general problem of missing covariates, time-varying covariates and so forth is discussed later in this chapter (section 11.6).

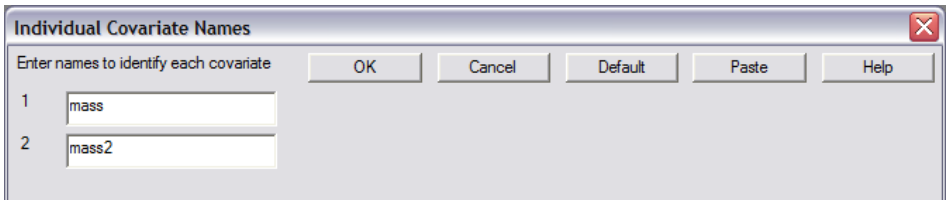
That's about it really, as far as data formatting goes. The next step involves bring these data into **MARK**, and specifying which covariates you want to use in your analyses, and how.

11.2.1. Specifying covariate data in MARK

Start program **MARK**, and begin a new project - '**`recaptures only`**'. We will use the data file `indcov1.inp`. This file has 8 occasions, and the data are in 'typical' mark-recapture '`LLLLL`' format. In this data file, are 2 covariates for each individual, which we'll call `mass` (for mass) and `mass2` (for mass^2). At this point, we need to 'tell' **MARK** we have 2 individual covariates (below):



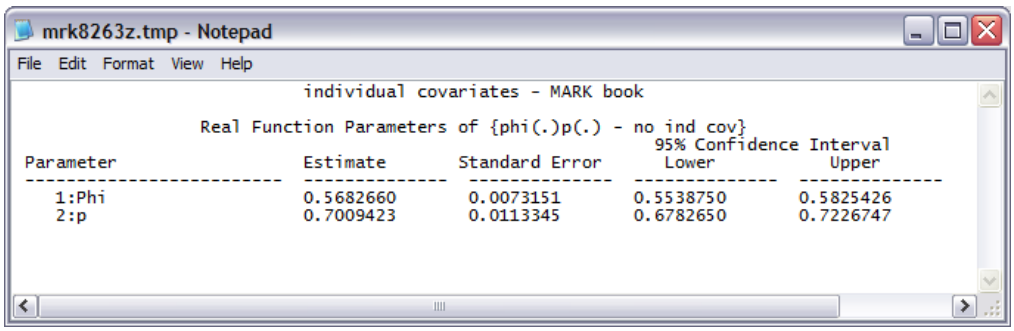
Next, we want to give the covariates some 'meaningful' names, so we click the '**`Enter Ind. Cov. Names`**' button. We'll use `mass` and `mass2` to refer to body mass and body mass-squared, respectively. That's it! From here on, we refer to the covariates in our analyses by using the assigned labels `mass` and `mass2`.



11.2.2. Executing the analysis

In this example, we simulated data with a constant survival and recapture probability over time. Thus, for our starting model, we will modify the model structure to reflect this - in other words, we'll start by fitting model $\{\varphi.\, p.\}$. Go ahead and set up this model using your preferred method (by either modifying the PIMs directly, or modifying the PIM chart), and run it. When you run **MARK**, you'll notice that it seems to take a bit longer to start the analysis. This is a result of the fact that this is a fairly large simulated data set, and that you are not using summary encounter histories - because we've told **MARK** that the data file contains individual covariates, **MARK** will build the likelihood piece by piece - or, rather, individual by individual. This process takes significantly longer than building the likelihood from data summarized over individuals.

Add the results to the browser. Let's have a look at the 2 reconstituted parameter estimates:



Start with parameter 2 - the recapture rate. The estimate of 0.7009 is very close to the 'true' value of $p = 0.70$ used in simulating the data (not surprising they should be so close given the size of the data set). What about the first parameter estimate - $\hat{\varphi} = 0.568$? This is the estimate of the apparent survival probability assuming (i) no time variation, and (ii) all individuals are the same. Clearly, it is this second assumption which is most important here, since we know (in this case) that all individuals in this data set are **not** the same - there is heterogeneity among individuals in survival rate, as a function of individual differences in body mass.

Thus, we expect that a model which accounts for this heterogeneity will fit significantly better than a model which ignores it. Where does the value of 0.568 come from? Remember that the actual probability of survival was set in the simulation to be a function of body mass:

$$\text{logit}(\varphi) = -0.039 + 0.0107(\texttt{mass}) - 0.000045(\texttt{mass}^2)$$

The data were simulated using a normal distribution with mean 110 mass units, and a standard deviation of 25. Thus, the value of 0.568 is the mean survival probability expected given the normal distribution of body mass values, and the function relating survival to body mass. However, if you put

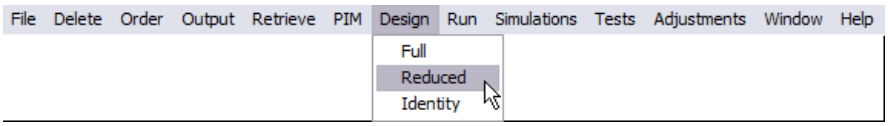
the value of '110' into this equation, you get an estimate of survival of $\hat{\varphi} = 0.594$, which is somewhat different from the reported value of $\hat{\varphi} = 0.568$. Why? Because what **MARK** is reporting is the mean survival of *the data set as a whole*: if you were to take *all* of the mass data in the input file, run each individual value for mass through the preceding equation, and take the mean of all of the generated values of φ, you would get an estimate of $\hat{\varphi} = 0.566$, which is basically identical to the value reported by **MARK**.

But, back to the question at hand - as suggested, we expect a model which incorporates individual covariates (body mass) to fit better than a model which ignores these differences. How do we go about fitting models with covariates?

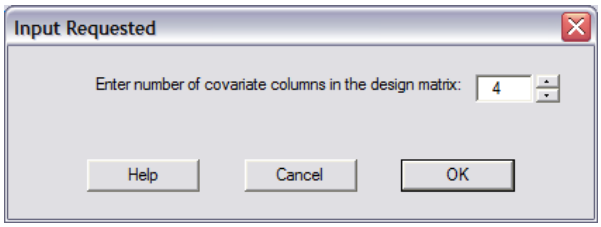
In fact, it's pretty easy, although depending on the model we want to fit, structuring the design matrix can require a bit of thought (*note*: 'a bit of thought' is always adaptive). In this case, we're effectively performing a multiple regression analysis. We want to take our starting model $\{\varphi_{.} p_{.}\}$ and constrain the estimates of survival to be functions of body mass, and (if we believe that normalizing selection is operating), the square of body mass. These were the 2 covariates contained in the input file (`mass` and `mass2`, respectively).

To fit a model with both `mass` and `mass2`, we need to modify the design matrix for our starting model. We can do this in several ways, but as a test of your understanding of the design matrix (discussed at length in Chapter 6), we'll consider it the following way. Our starting model is model $\{\varphi_{.} p_{.}\}$. One parameter for survival and recapture rate, respectively. Thus, the starting design matrix will be a (2×2) matrix. We want to modify this starting model to now include terms for `mass` and `mass2`. We want to constrain survival probability to be a function of both of these covariates. Remembering what you know about linear models and design matrices, you should recall that this means an intercept term, and one term ('slope') for `mass` and `mass2`, respectively. Thus, 3 terms in total, or, more specifically, 3 columns in the design matrix for survival, and 1 column for recapture rate.

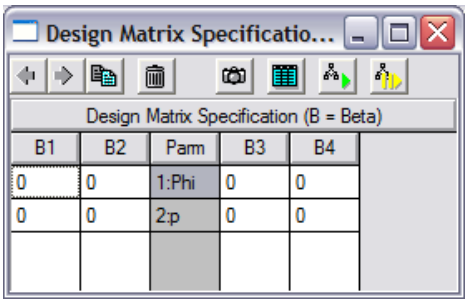
Let's look at how to do this. Select '**`design matrix | reduced`**'.



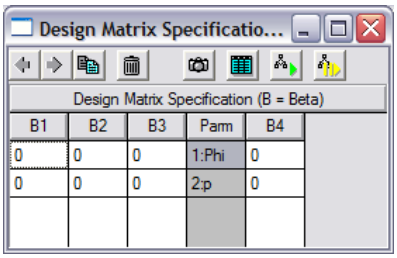
This will spawn a window asking you to specify the number of covariate columns you want. Translation - how many **total** columns do you want in your design matrix. As noted above, we want 4 columns - 3 to specify the survival parameter, and 1 to specify the encounter rate (since this is the parameter structure specified by the PIMs we created when we started). So, enter '4'.



Once you have entered the number of covariate columns you want in the design matrix, and clicked the '**OK**' button, you'll be presented with an 'empty' (4×2) design matrix.

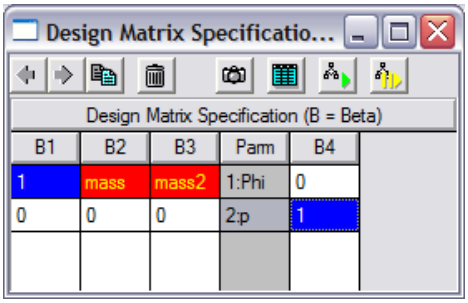


To start with, let's move the grey '**Parm**' column one column to the right, just to make things a bit clearer.



Now, all we need to do is add the appropriate values to the appropriate cells of the design matrix. If you remember any of the details from Chapter 6, you might at this moment be thinking in terms of '0' and '1' dummy variables. Well, you're not far off. We do more or less the same thing here, with one twist - we use the names of the covariates explicitly, rather than dummy variables, for those columns corresponding to the covariates.

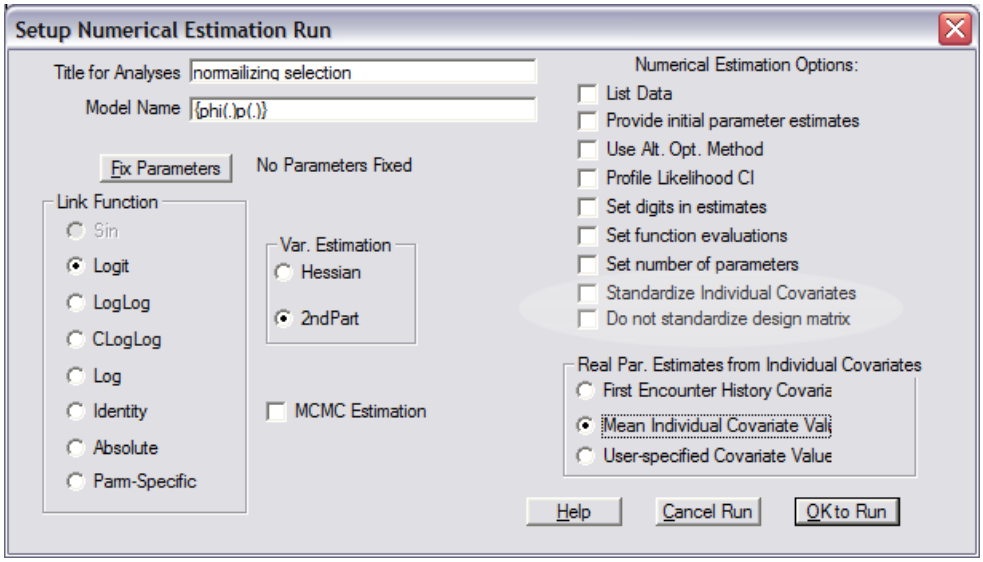
Let's start with survival rate. We have 3 columns in the design matrix to specify survival: 1 for the intercept, and 1 each for the covariates `mass` and `mass2`, respectively. For the intercept, we enter a '1' in the first cell of the first column. However, for the 2 covariate columns (columns 2 and 3), we enter the labels we assigned to the covariates, `mass` and `mass2`. For the recapture parameter, we simply enter a '1' in the lower right-hand corner. The completed design matrix for our model is shown below:



That's it! Go ahead and run this model. When you click on the '**Run**' icon, you'll be presented with

the '**Setup Numerical Estimation Run**' window. We need to give our model a title. We'll use '`phi(mass mass2)p(.)`' for the model specified by this design matrix. Again, notice that the sin link is no longer available – recall from Chapter 6 that the sin link is available only when the identity design matrix is used. The new 'default' is the logit link. We'll go ahead and use this particular link function.

Now, before we run the model, the first 'complication' of modeling individual covariates. On the right hand side of the '**Setup Numerical Estimation Run**' window, you'll notice a list of various options. Two of these options refer to 'standardizing' - the first, refers to standardizing the individual covariates. The second, specifies that you do not want to standardize the design matrix. These two '**standardization**' check boxes are followed by a nested list of suboptions (which have to do with how the real parameter estimates from the individual covariates are presented - more on this later).



The first check box (standardize individual covariates) essentially causes **MARK** to '*z-transform*' your individual covariates. In other words, take the value of the covariate for some individual, subtract from it the mean value of that covariate (calculated over all individuals), and divide by the standard deviation of the distribution of that covariate (again, calculated over all individuals). The end result is a distribution for the transformed covariate which has a mean of 0.0, and a standard deviation of 1.0, with individual transformed values ranging from approximately ($-3 \rightarrow +3$) (depending on the distribution of the individual data). One reason to standardize individual covariates in this way is to make all of your covariates have the same mean and variance, which can be useful for some purposes. Another reason is as an *ad hoc* method for accommodating any missing values in your data – if you use the z-transform standardization, the mean of the covariates over all individuals is 0, and thus missing data could simply be coded with 0 (which, again, is the mean of the transformed distribution). If you compute the mean of the non-missing values of an individual covariate, and then scale the non-missing values to have a mean of zero, the missing values can be included in the analysis as zero values, and will not affect the slope of the estimated β. However, this 'trick' is not advisable for a covariate with a large percentage of missing values because you will have little to no power. [The issue of 'missing values' is treated more generally in a later section of this chapter.] While these seem fairly reasonable and innocuous reasons to use this standardization option, there are several reasons to be very careful when using this option, as discussed in the following `-sidebar-`. In fact, it is because of some of these complications that the default condition for this standardization option is 'off'.

What about the second option - '**Do not standardize (the) design matrix**'? As noted in the **MARK** help file, it is often helpful to *scale* the values of the covariates to ensure that the numerical optimization algorithm finds the correct parameter estimates. The current version of **MARK** defaults to scaling your covariate data for you automatically (without you even being aware of it). This 'automatic scaling' is done by determining the maximum absolute value of the covariates, and then dividing each covariate by this value. This results in each column scaled to between -1 and 1. This internal scaling is purely for purposes of ensuring the success of the numerical optimization - the parameter values reported by **MARK** (i.e., in the output that you see) are 'back-transformed' to the original scale. There *may* be reasons you don't want **MARK** to perform this 'internal standardization' – if so, you simply check the '**Do not standardize (the) design matrix**' button.

begin sidebar

when to standardize - careful!

While using the z-transform standardization on your individual covariates may appear reasonable, or at the least, innocuous, you do need to think carefully about when, and how, to standardize individual covariates. For example, when you specify a model with a common intercept but 2 or more slopes for the individual covariate, and instruct **MARK** to standardize the individual covariate, you will get a different value of the deviance than from the model run with unstandardized individual covariates.

This behavior is because the centering effect of the standardization method affects the intercept differently depending on the value of the slope parameter. The effect is caused by the nonlinearity of the logit link function. You get the same effect if you standardize variables in a logistic regression, and run them with a common intercept. The result is that the estimates are not scale independent, but depend on how much centering is performed by subtracting the mean value. In other words, situations can arise where the real parameter estimates and the model's AIC differ between runs using the standardized covariates and the unstandardized covariates. This situation arises because the z-transformation affects both the slope and intercept of the model. For example, with a logit link function and the covariate x_1,

$$\begin{aligned}\text{logit}(S) &= \beta_1 + \beta_2 \left(x_1 - \bar{x_1}\right)/SD_1 \\ &= \left(\beta_1 - \beta_2 \bar{x_1}/SD_1\right) + (\beta_2/SD_1)x_1\end{aligned}$$

where the intercept is the quantity shown in the first set of brackets, and the second bracket is the slope. This result shows the conversion between the β parameter estimates for the standardized covariate and the β parameter estimates for the untransformed covariate, i.e., the intercept for the untransformed analysis would correspond to the quantity in the first set of brackets, and the slope for the untransformed analysis would correspond to the quantity in the second set of brackets. All well and good so far, because the model with a standardized covariate and the model with the unstandardized covariate will result in identical models with identical AIC_c values.

However, now consider the case where we have 2 groups, and want to build a model with different slope parameters for each group's individual covariate values, but a common intercept. In this example, x_1 and x_2 are considered to be the same individual covariate, each standardized to the overall mean and SD, but with values specific to group 1 (x_1) or group 2 (x_2). The *unstandardized* model would look like:

Group 1: $\text{logit}(S_1) = \beta_1 + \beta_2 x_1$

Group 2: $\text{logit}(S_2) = \beta_1 + \beta_3 x_2$

Unfortunately, when the individual covariates *are* standardized, the result is:

Group 1: $\text{logit}(S_1) = (\beta_0 - \beta_1 \bar{x}_1/SD) + (\beta_1/SD)x_1$

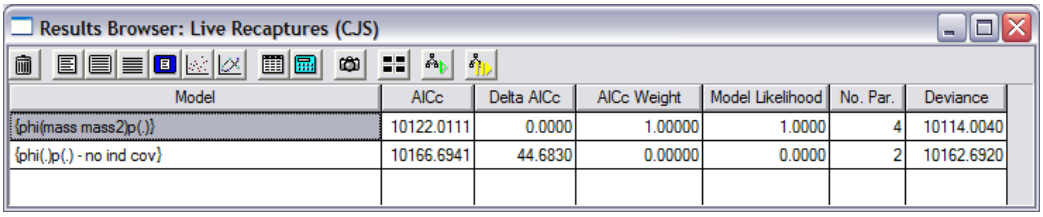
Group 2: $\text{logit}(S_2) = (\beta_0 - \beta_2 \bar{x}_2/SD) + (\beta_2/SD)x_2$

In this case, the intercepts for the 2 groups are no longer the same with the standardized covariates, resulting in a different model with a different AIC_c value than for the unstandardized case. This difference causes the AIC values for the 2 models to differ because the real parameter estimates differ between the 2 models.

An alternative to this *z*-transformation is to use the `product` function in the design matrix (c.f. p. 20) to multiply the individual covariate by a *scaling* value. As an example, suppose the individual covariate `Var` ranges from 100 to 900. Using the design matrix function `product(Var,0.001)` in the entries of the design matrix would result in values ranging from 0.1 to 0.9, and would result in 3 more significant digits being reported in the estimates of the β parameter for this individual covariate.

end sidebar

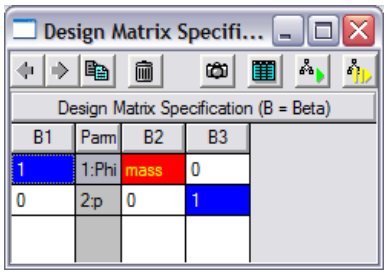
Acknowledging the need for caution discussed in the preceding -sidebar-, for purposes of demonstration, we'll go ahead and run our model, using the *z*-transformation on the covariate data (by checking the '**`Standardize Individual Covariates`**' checkbox). Add the results to the browser.



Model	AICc	Delta AICc	AICc Weight	Model Likelihood	No. Par.	Deviance
{phi(mass mass2)p(.)}	10122.0111	0.0000	1.00000	1.0000	4	10114.0040
{phi(.)p(.) - no ind cov}	10166.6941	44.6830	0.00000	0.0000	2	10162.6920

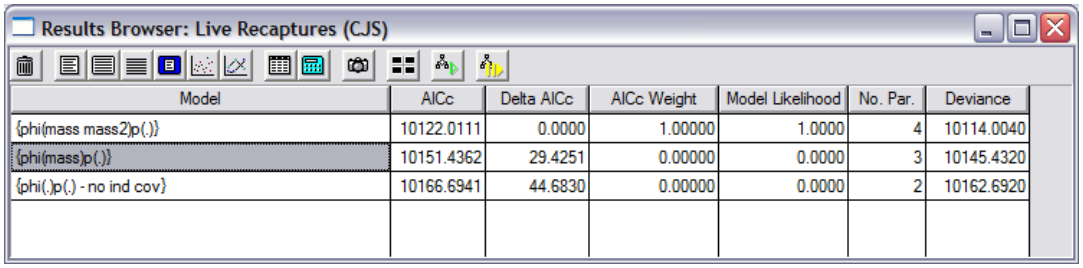
First, we notice right away that the model including the 2 covariates fits **much** better than the model which doesn't include them - so much so that it is clear there is effectively no support for our naïve starting model.

Do we have any evidence to support our hypothesis that there is normalizing selection on body mass? Well, to test this, we might first want to run a model which does not include the `mass2` term. Recall that it was the inclusion of this second order term which allowed for a decrease in survival with mass beyond some threshold value. How do you run the model with `mass`, but not `mass2`? The easiest way to do this is to simply eliminate the column corresponding to `mass2` from the design matrix. So, simply bring the design matrix for the current model up on the screen (by retrieving the current model), and delete the column corresponding to `mass2` (i.e., delete column 3 from the design matrix). The modified design matrix now looks like:



B1	Parm	B2	B3
1	1:Phi	mass	0
0	2:p	0	1

Go ahead and run this model - again using standardized covariates. Call this model '`phi(mass)p(.)`'. Add the results to the results browser.


Results Browser: Live Recaptures (CJS)

Model	AICc	Delta AICc	AICc Weight	Model Likelihood	No. Par.	Deviance
{phi(mass mass2)p(.)}	10122.0111	0.0000	1.00000	1.0000	4	10114.0040
{phi(mass)p(.)}	10151.4362	29.4251	0.00000	0.0000	3	10145.4320
{phi(.)p(.) - no ind cov}	10166.6941	44.6830	0.00000	0.0000	2	10162.6920

Note that the model with `mass` only (but not the second order term) fits better than our general starting model, but nowhere near as well as the model including both `mass` and `mass2` - it has essentially no support. In other words, our model with both `mass` and `mass2` is clearly the best model for these data (this is not surprising, since this is the very model we used to simulate the data in the first place!).

So, at this stage, we could say with some assurance that there is fairly strong support for the hypothesis that there is normalizing selection on body mass. However, suppose we want to actually look at the 'shape' of this function. How can we derive the function relating survival to mass, given the results from our **MARK**? In fact, it's fairly easy, *if you remember the details concerning the logit transform, and how we standardized our data*.

To start, let's look at the output from **MARK** for the model including `mass` and `mass2` (shown at the top of the next page). In this case, it's easier to use the '**`full results`**' option (i.e., the option in the browser toolbar which presents all of the details of the numerical estimation). Scroll down until you come to the section shown at the top of the next page. Note that we have 3 sections of the output at this point. In the first section we see the estimated logit function parameters for the model. There are 4 β values, corresponding to the 4 columns of the design matrix (the `intercept`, `mass`, `mass2` and the encounter rate, `p`, respectively). These parameters, in fact, are what we need to specify the function relating survival to body weight.

In fact, if you think about it, only the first 3 of these logit parameters are needed - the last one refers to the encounter probabiliity, which is not a function of body mass. What is our function? Well, it is

$$\text{logit}(\hat{\varphi}) = 0.256733 + 1.1750545(\texttt{mass}_s) - 1.0555046(\texttt{mass}_s^2)$$

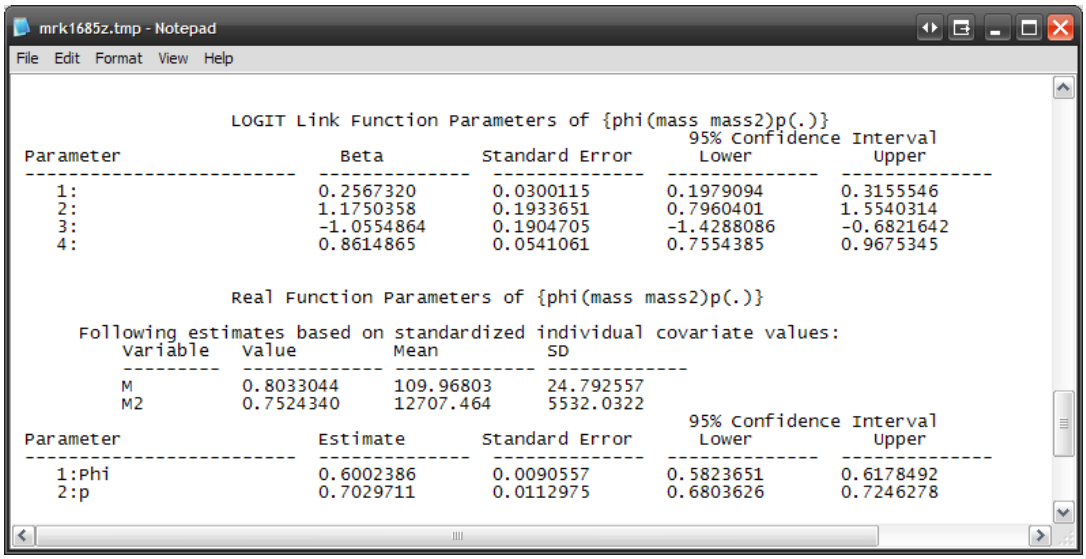
Note that for the two mass terms, we have added a small subscript '*s*' - reflecting the fact that these are 'standardized' masses. Recall that we standardized the covariates by subtracting the mean of the covariate, and dividing by the standard deviation. Thus, for each individual,

$$\text{logit}(\hat{\varphi}) = 0.256733 + 1.17505\left(\frac{\texttt{m} - \bar{\texttt{m}}}{\text{SD}_{\texttt{m}}}\right) - 1.0555\left(\frac{\texttt{m}^2 - \bar{\texttt{m}}^2}{\text{SD}_{\texttt{m}^2}}\right)$$

In this expression, `m` refers to `mass` and $\texttt{m}^2$ refers to `mass2`.

The output from **MARK** (shown at the top of the next page) actually gives you the mean and standard deviations for both covariates. For `mass`, mean = 109.97, and SD = 24.79, while for `mass2`, the mean = 12,707.46, and the SD = 5,532.03. The 'value' column shows the standardized values for `mass` and `mass2` (0.803 and 0.752) for the first individual in the data file. Let's look at an example. Suppose the mass of the bird was 110 units. Thus `mass` = 110, `mass2` = 110^2 = 12,100. Thus,

$$\text{logit}(\hat{\varphi}) = 0.2567 + 1.17505\left(\frac{110 - 109.97}{24.79}\right) - 1.0555\left(\frac{12{,}100 - 12{,}707.46}{5{,}532.03}\right) = 0.374.$$



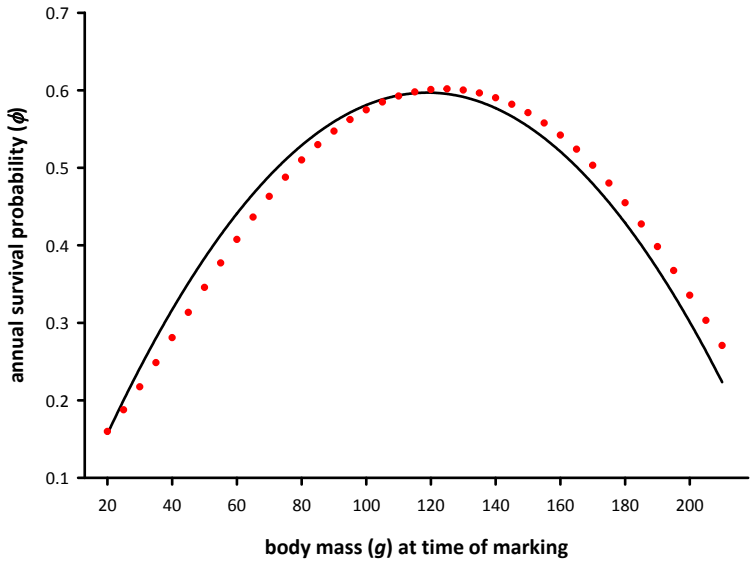
So, if $\text{logit}(\hat{\varphi}) = 0.374$, then how do we get the reconstituted values for survival? Recall that

$$\text{logit}(\theta) = \log\left(\frac{\theta}{1-\theta}\right) = \alpha + \beta x$$

and

$$\theta = \frac{e^{\alpha+\beta x}}{1+e^{\alpha+\beta x}}$$

Thus, if $\text{logit}(\hat{\varphi}) = 0.374$, then the reconstituted estimate of φ, transformed back from the logit scale is $e^{0.374}/(1 + e^{0.374}) = 0.592$. Thus, for an individual weighing 110 units, expected annual survival probability is 0.592. How well does the estimated function match with the 'true' function used to simulate the data? Let's plot the observed versus expected values:



As you can see from the plot, the fit between the values expected given the 'true' function (solid black line) and those based on the function estimated from **MARK** (red dots) are quite close, as they should be. The slight deviation between the two is simply because the simulated data are simply one realization of the stochastic process governed by the underlying survival and recapture parameters.

Note: in the preceding, we've described the mechanics of reconstituting the parameter estimate - this basically involves back-transforming from the logit scale to the normal [0, 1] probability scale. What about reconstituting the variance, or SE of the estimate, on the normal scale? This is somewhat more complicated. As briefly introduced in Chapter 6, reconstituting the sampling variance on the normal scale involves use of something known as the '*Delta method*'. The Delta method, and its application to reconstituting estimates of sampling variance (both generally, and for models involving individual covariates) is discussed at length in Appendix B.

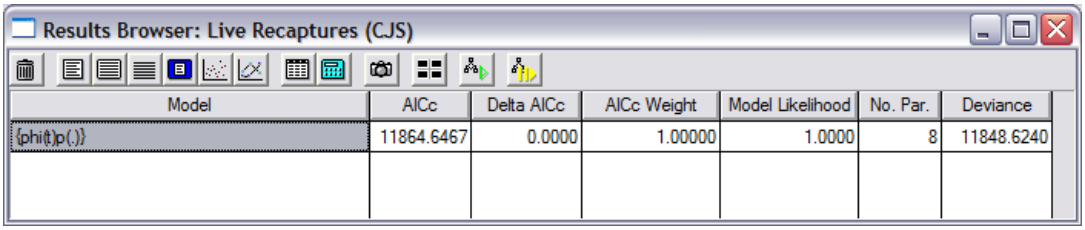
11.3. A more complex example - time variation

In the preceding example, we made life simple by simulating some data where there was no variation in either survival or recapture rates over time. In this example, we'll consider the more complicated problem of handling data where there is potential variation in survival over time.

We'll use the same approach as before, except this time we will simulate some data where survival probability is a complex function of both mass and cohort. In this case, we simulated a data set having normalizing selection in early cohorts, with a progressive shift towards diversifying selection in later cohorts. Arguably, this is a rather 'artificial' example, but it will suffice to demonstrate some of the considerations involved in using **MARK** to handle temporal variation in the relationship between estimates of one or more parameters and one or more individual covariates.

The data for this example are contained in `indcov2.inp`. We simulated 8 occasions, and assumed a constant recapture rate ($p = 0.7$) for all individuals in all years. The data file contains 2 covariates - `mass` and `mass2` (as in the previous example). As with the first example, we start by creating a new project, and importing the `indcov2.inp` data file. Label the two covariates `mass` and `mass2` (respectively).

We will start by fitting model $\{\varphi_t p_.\}$, since this is structurally consistent with the data, and will provide a reasonable starting point for comparisons with subsequent models. Go ahead and add the results of this model to the results browser.

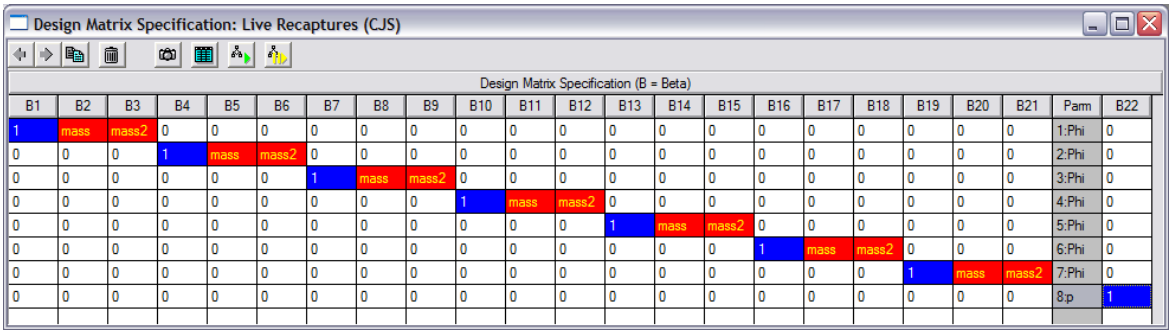


Model	AICc	Delta AICc	AICc Weight	Model Likelihood	No. Par.	Deviance
{phi(t)p(.)}	11864.6467	0.0000	1.00000	1.0000	8	11848.6240

Now, to fit models with both individual covariates, and time variation in the relationship between survival and the covariates, we need to think a bit more carefully than in our first example. If you understood the first example, you might realize that to do this, we need to modify the design matrix. However, how we do this will depend on what hypothesis we want to test. For example, we might believe that the relationship between survival and mass changes with each time interval. Alternatively, we might suppose there is a common intercept, but different slopes for each interval. It is important to consider carefully what hypothesis you want to test before proceeding.

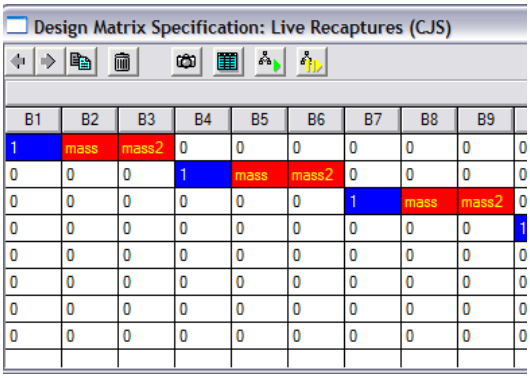
We'll start with the hypothesis that the relationship between survival and mass changes with each time interval. With a bit of thought, you might guess how to construct this design matrix. In the previous example, we used 3 columns to specify this relationship - representing the intercept, `mass` and `mass2`, respectively. However, in the first example, we assumed that this model was constant over all years. So, what do we do if we believe the relationship varies from year to year? Easy, we simply have 3 columns for each interval in the design matrix for survival (with 1 additional column at the end for the constant recapture rate). So, 7 intervals = 21 columns for survival, plus 1 column for the recapture rate. How many rows? Remembering from Chapter 6, 8 rows total - 7 rows for the 7 survival intervals, and 1 for the constant recapture rate.

So, let's go ahead and construct the design matrix for this model, using the '**`Design | Reduced`**' menu option we discussed previously. We'll start simply, using a DM based on the basic structure of the *identity* matrix - recall that for an identity DM, each row corresponds to a 'time-specific regression model', since each row has its own intercept (see Chapter 6). This matrix (shown below) is sufficiently big such that it's rather difficult to see the entire structure at once.



B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	B11	B12	B13	B14	B15	B16	B17	B18	B19	B20	B21	Parm	B22
1	mass	mass2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1:Phi	0
0	0	0	1	mass	mass2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2:Phi	0
0	0	0	0	0	0	1	mass	mass2	0	0	0	0	0	0	0	0	0	0	0	0	3:Phi	0
0	0	0	0	0	0	0	0	0	1	mass	mass2	0	0	0	0	0	0	0	0	0	4:Phi	0
0	0	0	0	0	0	0	0	0	0	0	0	1	mass	mass2	0	0	0	0	0	0	5:Phi	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	mass	mass2	0	0	0	6:Phi	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	mass	mass2	7:Phi	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8:p	1

To help you visualize it, let's look at just a small piece of this design matrix:



B1	B2	B3	B4	B5	B6	B7	B8	B9
1	mass	mass2	0	0	0	0	0	0
0	0	0	1	mass	mass2	0	0	0
0	0	0	0	0	0	1	mass	mass2
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0

As you can see, for each survival interval, we have 3 columns - 1 intercept, and 1 column each for `mass` and `mass2`, respectively. So, the columns `B1`, `B2` and `B3` correspond to interval 1, `B4`, `B5` and `B6` for interval 2, and so on. You simply do this for each of the 7 survival intervals. The bottom right-hand cell of the matrix (shown on the preceding page) contains a single '1' for the constant encounter probability. Call this model '`phi(t * mass mass2)p(.)`', and run it - remember to standardize the covariates before running the model. Add the results to the browser.

Results Browser: Live Recaptures (CJS)

Model	AICc	Delta AICc	AICc Weight	Model Likelihood	No. Par.	Deviance
{phi(t*mass mass2)p(.)}	11809.5367	0.0000	1.00000	1.0000	22	11765.3770
{phi(t)p(.)}	11864.6467	55.1100	0.00000	0.0000	8	11848.6240

Again, note that the model constrained to be a function of `mass` and `mass2` fits much better than our naïve starting model. Let's test this model against one which does not have time variation in the relationship between survival and body mass. Now, there are at least 2 ways to do this. First, we could take our design matrix, and reduce it to only 4 columns:

Design Matrix Specific...

Design Matrix Specification (B = Beta)

B1	B2	Parm	B3	B4
1	mass	1:Phi	mass2	0
1	mass	2:Phi	mass2	0
1	mass	3:Phi	mass2	0
1	mass	4:Phi	mass2	0
1	mass	5:Phi	mass2	0
1	mass	6:Phi	mass2	0
1	mass	7:Phi	mass2	0
0	0	8:p	0	1

In this case, we're imposing the same model on each of the survival estimates for each of the 7 intervals. In other words, we're constraining model $\{\varphi_t p_.\}$ using the same model in each year.

However, as a test of your understanding, recall from the first example that we could achieve exactly the same result if we (i) start with model $\{\varphi_. p_.\}$ and then (ii) constrain the estimates of survival to be a function of `mass` and `mass2`.

Design Matrix Specificatio...

Design Matrix Specification (B = Beta)

B1	B2	B3	Parm	B4
1	mass	mass2	1:Phi	0
0	0	0	2:p	1

If you've been paying attention, you may have noticed that this is **exactly** what we did in the first example – these 2 design matrices yield exactly the same results. Pick whichever one is easiest - we'll use the first approach, and call the model '`phi(mass mass2)p(.)`'. Add the results to the browser.

Results Browser: Live Recaptures (CJS)

Model	AICc	Delta AICc	AICc Weight	Model Likelihood	No. Par.	Deviance
{phi(t*mass mass2)p(.)}	11809.5367	0.0000	1.00000	1.0000	22	11765.3770
{phi(t)p(.)}	11864.6467	55.1100	0.00000	0.0000	8	11848.6240
{phi(mass mass2)p(.)}	12077.3083	267.7716	0.00000	0.0000	4	12069.3020

Note that this model does not fit well at all - even worse than our naïve starting model. This is not surprising, since there is clearly annual variation in our estimates of survival - after all, we simulated the data that way! Also note that only 4 parameters are estimated. Remember, the number of columns – **not** the number of rows – in the design matrix determines the number of parameters that are potentially estimated.

Suppose you want to test the hypothesis that there is a common intercept for each year, but a different slope. How would you modify the design matrix to show this? Well, by now you might have guessed - you simply have 1 column for an intercept for all 7 intervals, and then multiple columns for the `mass` and `mass2` terms for each interval:

Design Matrix Specification: Live Recaptures (CJS)

Design Matrix Specification (B = Beta)

B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	B11	B12	B13	B14	B15	Parm	B16
1	mass	mass2	0	0	0	0	0	0	0	0	0	0	0	0	1:Phi	0
1	0	0	mass	mass2	0	0	0	0	0	0	0	0	0	0	2:Phi	0
1	0	0	0	0	mass	mass2	0	0	0	0	0	0	0	0	3:Phi	0
1	0	0	0	0	0	0	mass	mass2	0	0	0	0	0	0	4:Phi	0
1	0	0	0	0	0	0	0	0	mass	mass2	0	0	0	0	5:Phi	0
1	0	0	0	0	0	0	0	0	0	0	mass	mass2	0	0	6:Phi	0
1	0	0	0	0	0	0	0	0	0	0	0	0	mass	mass2	7:Phi	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8:p	1

Call this model '`phi(t mass mass2 - common intercept,p(.)`', and run it. Again, this model takes some time to run. Once it's finished, add the results to the browser.

Results Browser: Live Recaptures (CJS)

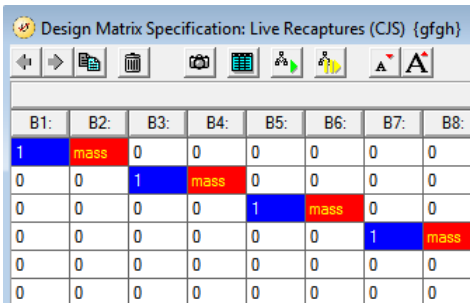
Model	AICc	Delta AICc	AICc Weight	Model Likelihood	No. Par.	Deviance
{phi(t*mass mass2)p(.)}	11809.5367	0.0000	1.00000	1.0000	22	11765.3770
{phi(t)p(.)}	11864.6467	55.1100	0.00000	0.0000	8	11848.6240
{Phi(t mass mass2 - common intercept,p(.)}	12060.9498	251.4131	0.00000	0.0000	16	12028.8640
{phi(mass mass2)p(.)}	12077.3083	267.7716	0.00000	0.0000	4	12069.3020

Note that this model does not fit the data well at all - not surprising since the data were simulated under a model where the intercept varies significantly from year to year. Reconstituting estimates of survival from the estimated function in this example proceeds in the same fashion as in the first example - the only complication is that more 'book-keeping' is required. Each year has its own function. If you look at the parameter estimates from our most parsimonious model, you will see that they correspond to what we expected: in the early cohorts the sign of the slope for `mass` is positive, and for `mass2` is negative - consistent with normalizing selection. In later cohorts, the signs are consistent with increasingly disruptive selection.

It is worth noting that when you specify a model with a common intercept but 2 or more slopes for the individual covariate, and standardize the individual covariate, you will get a different value of the deviance than from the model run with unstandardized individual covariates. This is because the centering effect of the standardization method affects the intercept differently depending on the value of the slope parameter. The effect is caused by the nonlinearity of the logit link function. You get the same effect if you standardize variables in a logistic regression, and run them with a common intercept. The result is that the estimates are not scale independent, but depend on how much centering is performed by subtracting the mean value.

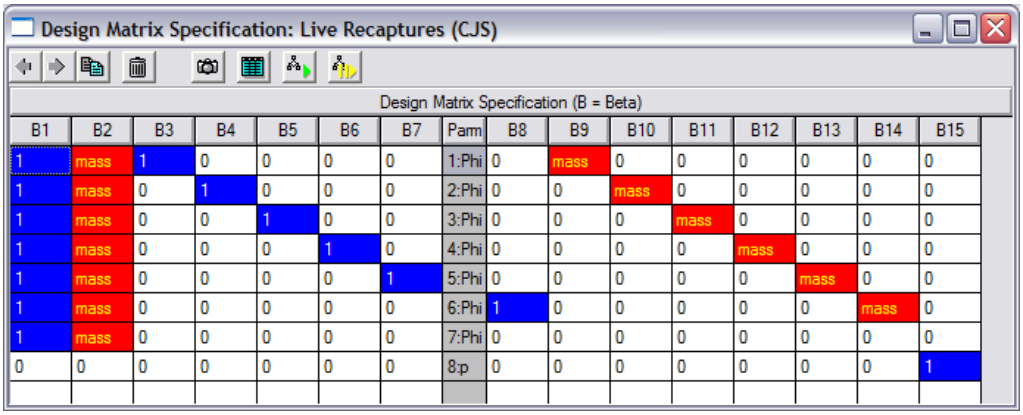
11.4. The DM & individual covariates - some elaborations

Suppose you want to fit a model with different intercepts and different slopes for each year. As we've already seen, you can accomplish this by adding an 'intercept' and 'slope' parameters to each row for the parameter in question. In our example where we modeled survival to be a function of body mass, we might have used:


Design Matrix Specification: Live Recaptures (CJS) {gfgh}

B1:	B2:	B3:	B4:	B5:	B6:	B7:	B8:
1	mass	0	0	0	0	0	0
0	0	1	mass	0	0	0	0
0	0	0	0	1	mass	0	0
0	0	0	0	0	0	1	mass
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

However, a more flexible way to model this would have been to use:


Design Matrix Specification: Live Recaptures (CJS)

Design Matrix Specification (B = Beta)

B1	B2	B3	B4	B5	B6	B7	Parm	B8	B9	B10	B11	B12	B13	B14	B15
1	mass	1	0	0	0	0	1:Phi	0	mass	0	0	0	0	0	0
1	mass	0	1	0	0	0	2:Phi	0	0	mass	0	0	0	0	0
1	mass	0	0	1	0	0	3:Phi	0	0	0	mass	0	0	0	0
1	mass	0	0	0	1	0	4:Phi	0	0	0	0	mass	0	0	0
1	mass	0	0	0	0	1	5:Phi	0	0	0	0	0	mass	0	0
1	mass	0	0	0	0	0	6:Phi	1	0	0	0	0	0	mass	0
1	mass	0	0	0	0	0	7:Phi	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	8:p	0	0	0	0	0	0	0	1

In other words, a column of '1's for the intercept, a column for the covariate (mass), and then the columns of dummy variables corresponding to each of the time intervals, and then columns reflecting the interaction of the the covariate and time. You might recognize this as the same analysis of covariance (ANCOVA) design you saw back in Chapter 6. If you take this design matrix, and run it, you'll see that you get exactly the same results as you did with the design matrix we used initially - each leads to time-specific estimates of the slope and intercept.

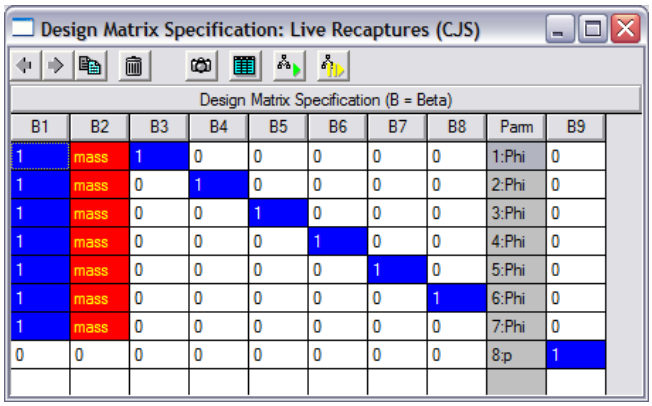
So, if they both yield the 'same results', why even consider this more formal design matrix? As we noted in Chapter 6, the biggest advantage is that using this more complete (formal) design matrix allows you to test some models which aren't possible using the first approach. Specifically, in this case, the additive model - different intercept, but common slope. In other words, testing model

$$\varphi = \texttt{time} + \texttt{mass}$$

in addition to

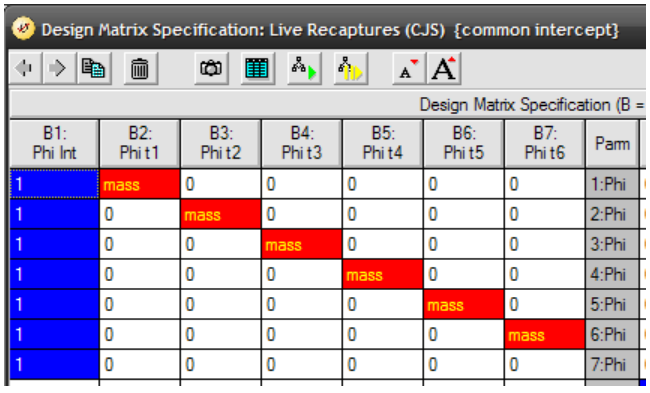
$$\varphi = \texttt{time} + \texttt{mass} + \texttt{time.mass}$$

As we discussed in Chapter 6, this sort of additive model can only be fit using this formal design-matrix approach. So, to fit this model – where we have different intercepts, but a common slope among years – we simply delete the interaction columns. It's that simple! Here is the reduced design matrix:



B1	B2	B3	B4	B5	B6	B7	B8	Parm	B9
1	mass	1	0	0	0	0	0	1:Phi	0
1	mass	0	1	0	0	0	0	2:Phi	0
1	mass	0	0	1	0	0	0	3:Phi	0
1	mass	0	0	0	1	0	0	4:Phi	0
1	mass	0	0	0	0	1	0	5:Phi	0
1	mass	0	0	0	0	0	1	6:Phi	0
1	mass	0	0	0	0	0	0	7:Phi	0
0	0	0	0	0	0	0	0	8:p	1

If instead you wanted a common intercept for all years, but different slopes for `mass` for each year, then the DM would look like:



B1: Phi Int	B2: Phi t1	B3: Phi t2	B4: Phi t3	B5: Phi t4	B6: Phi t5	B7: Phi t6	Parm
1	mass	0	0	0	0	0	1:Phi
1	0	mass	0	0	0	0	2:Phi
1	0	0	mass	0	0	0	3:Phi
1	0	0	0	mass	0	0	4:Phi
1	0	0	0	0	mass	0	5:Phi
1	0	0	0	0	0	mass	6:Phi
1	0	0	0	0	0	0	7:Phi

In general, our recommendation is to use the full design matrix as much as possible. While it is admittedly tedious to construct them, especially for complicated designs, it does give you ultimate flexibility in constructing models. It also provides a formal, common frame of reference to help you understand what model you're actually fitting!

begin sidebar

Design Matrix Functions

A number of special functions are allowed as entries in the design matrix: `add`, `product`, `power`, `min`, `max`, `log`, `exp`, `eq` (equal to), `gt` (greater than), `ge` (greater than or equal to), `lt` (less than), and `le` (less than or equal to). These names can be either upper- or lower-case. You should not include blanks within these function specifications to allow **MARK** to properly retrieve models with these functions in their design matrix. As shown below, these functions can be nested to create quite complicated expressions, which may require setting a larger value of the design matrix cell size (something you can specify by changing **MARK**'s preferences - '**`File | Preferences`**').

1. `add` and `product` functions

These two functions require 2 arguments. The `add` function adds the 2 arguments together, whereas the product function multiplies the 2 arguments. The arguments for both functions must be one of the 3 types allowed: numeric constant, an individual covariate, or another function call. The following design matrix demonstrates the functionality of these 2 functions, where `wt` is an individual covariate.

```
1   1      1       wt   product(1,wt)   product(wt,wt)
1   1      2       wt   product(2,wt)   product(wt,wt)
1   1      3       wt   product(3,wt)   product(wt,wt)
1   0   add(0,1)   wt   product(1,wt)   product(wt,wt)
1   0   add(1,1)   wt   product(2,wt)   product(wt,wt)
1   0   add(1,2)   wt   product(3,wt)   product(wt,wt)
```

The use of the `add` function in column 3 is just to demonstrate examples; it would not be used in a normal application. In each case, a continuous variable is created by adding constant values. The results are the values 1, 2, and 3, in rows 4, 5, and 6, respectively.

Column 5 of the design matrix demonstrates creating an interaction between an individual covariate and another column (the first 3 rows) or a constant and an individual covariate (the last 3 rows). Column 6 of the design matrix demonstrates creating a quadratic effect for an individual covariate. Note that if the 2 arguments were different individual covariates, an interaction effect between 2 individual covariates would be created in column 6.

2. IF functions: `eq` (equal to), `gt` (greater than), `ge` (greater than or equal to), `lt` (less than), `le` (less than or equal to)

These five functions require 2 arguments. The `eq`, `gt`, `ge`, `lt`, and `le` functions will return a zero if the operation is false and a one if the operation is true. For each of these functions, 2 arguments (`x1` and `x2`) are compared based on the function. For example, `eq(x1,x2)` returns 1 if `x1` equals `x2`, and zero otherwise; `gt(x1,x2)` returns 1 if `x1` is greater than `x2`, zero otherwise; and `le(x1,x2)` returns 1 if `x1` is less than or equal to `x2`, zero otherwise. The arguments for these functions must be one of the 3 types allowed: numeric constant, column variable, or an individual covariate. The following design matrix demonstrates the functionality of both the add function and the `IF` function (`eq`), where `age` is an individual covariate.

```
1   add(0,age)   eq(0,add(0,age))
1   add(1,age)   eq(0,add(1,age))
1   add(2,age)   eq(0,add(2,age))
1   add(3,age)   eq(0,add(3,age))
1   add(4,age)   eq(0,add(4,age))
1   add(5,age)   eq(0,add(5,age))
```

In this particular example, the individual covariate `age` corresponds to the number of days before a bird fledges from its nest (fledge day 0) and subsequently enters the study. Suppose an individual

fledges from its nest during the fourth survival period. Its encounter history (LDLD format) would consist of '00 00 00 10' and the individual would have -3 as its age covariate because the individual did not fledge from its nest until the fourth survival period. A bird that did not fledge from its nest until survival period 20 would have -19 as its age covariate. Think of the use of negative numbers as an accounting technique to help identify when the individual fledges.

Column 2 of the design matrix demonstrates the use of the add function to create a continuous age covariate for each individual by adding a constant to age. The value returned in the first row of the second column is -3 (0 + -3 = -3). The value returned in the second row of the second column is -2 (1 + -3 = -2). The value returned in the fourth row of the second column is zero and corresponds to fledge day 0 (3 + -3 = 0). The value returned in the fifth row of the second column is one and corresponds to fledge day 1. Thus, column 2 is producing a trend effect of age on survival, with the intercept of the trend model being age zero. A trend model therefore models a constant rate of change with age on the logit scale, so that each increase in age results in a constant change in survival, either positive or negative depending on the sign of β_2.

Now, suppose that survival is thought to be different on the first day that a bird fledges, i.e., the first day that the bird enters the encounter history. To model survival as a function of fledge day 0, use the eq function to create the necessary dummy variable. This is demonstrated in the third column. The eq function returns a value of one only when the statement is true, which only occurs on the first day the bird is fledged. Recall that the value for age of this individual is -3; therefore, the add function column will return a value of -3 (0 + -3 = -3) in the first row. The eq function in the third column would return a value of zero because age (-3) is not equal to zero. The eq function in the third column, fourth row would return a value of one because age (0) is equal to (0). Note this will only be true for row four for this particular individual; all other rows return a value of zero because they are false. Thus, the eq function will produce a dummy variable allowing for a different survival rate on the first day after fledging from the trend model for age which applies thereafter.

Note that the eq function in this example is using the same results of the add function from the preceding column, and illustrates the nesting of functions.

3. power function

This function requires 2 arguments (x,y). The first argument is raised to the power of the second argument; i.e., the result is x^y. As an example, to create a squared term of the individual covariate length, you would use power(length,2). To create a cubic term, power(length,3). So, in our normalizing selection example (first example of this chapter), we did not need to explicitly include mass^2 in the .INP file - we could have used power(mass,2) to accomplish the same thing.

4. min/max functions

The min function returns the minimum of the 2 arguments, whereas the max function returns the maximum of the 2 arguments. These functions allow the creation of thresholds with individual covariates. So, with the individual covariate length, the function min(5,length) would use the value of length when the variable is < 5, but replace length with the value 5 for all lengths > 5. Similarly, max(3,length) would replace all lengths < 3 with the value 3.

5. Log, Exp functions

These functions are equivalent to the natural logarithm function and the exponential function. Each only requires one argument. So, for the individual covariate length = 2, log(length) returns 0.693147181, and exp(length) returns 7.389056099.

Example

These functions are useful for constructing a design matrix when using the nest survival analysis (Chapter 17). Here, the add and ge functions are demonstrated. Stage-specific survival (egg or nestling) could be estimated only if nests were aged and frequent nest checks were done to assess stage of failure.

```
1   add(0,age)   GE(add(0,age),15)    product(add(0,age),GE(add(0,age),15))
1   add(1,age)   GE(add(1,age),15)    product(add(1,age),GE(add(1,age),15))
1   add(2,age)   GE(add(2,age),15)    product(add(2,age),GE(add(2,age),15))
1   add(3,age)   GE(add(3,age),15)    product(add(3,age),GE(add(3,age),15))
1   add(4,age)   GE(add(4,age),15)    product(add(4,age),GE(add(4,age),15))
1   add(5,age)   GE(add(5,age),15)    product(add(5,age),GE(add(5,age),15))
1   add(6,age)   GE(add(6,age),15)    product(add(6,age),GE(add(6,age),15))
1   add(7,age)   GE(add(7,age),15)    product(add(7,age),GE(add(7,age),15))
1   add(8,age)   GE(add(8,age),15)    product(add(8,age),GE(add(8,age),15))
1   add(9,age)   GE(add(9,age),15)    product(add(9,age),GE(add(9,age),15))
1   add(10,age)  GE(add(10,age),15)   product(add(10,age),GE(add(10,age),15))
1   add(11,age)  GE(add(11,age),15)   product(add(11,age),GE(add(11,age),15))
1   add(12,age)  GE(add(12,age),15)   product(add(12,age),GE(add(12,age),15))
1   add(13,age)  GE(add(13,age),15)   product(add(13,age),GE(add(13,age),15))
1   add(14,age)  GE(add(14,age),15)   product(add(14,age),GE(add(14,age),15))
1   add(15,age)  GE(add(15,age),15)   product(add(15,age),GE(add(15,age),15))
1   add(16,age)  GE(add(16,age),15)   product(add(16,age),GE(add(16,age),15))
1   add(17,age)  GE(add(17,age),15)   product(add(17,age),GE(add(17,age),15))
1   add(18,age)  GE(add(18,age),15)   product(add(18,age),GE(add(18,age),15))
```

In this particular example, the age covariate corresponds to the day that the first egg was laid in a nest (nest day 0). Suppose a nest is initiated during the fourth survival period. Its encounter history (LDLD format) would consist of `00 00 00 10` and the nest would have -3 as its age covariate because the first egg was not laid in the nest until the fourth survival period.

Column 2 of the design matrix demonstrates the use of the add function to create a continuous age covariate for each nest. The value returned in the first row of the second column is -3. The value returned in the second row of the second column is -2. The value returned in the fourth row of the second column is a zero and corresponds to the initiation of egg laying. The value returned in the fifth row of the second column is one (the nest is one day old).

To model survival as a function of stage, use the ge function to quickly create the necessary dummy variable. This is demonstrated in third column. The value of 15 is used in this example because it corresponds to the number of days before a nest will hatch young birds. Day 0 begins with the laying of the first egg, so values of 0-14 correspond to the egg stage. Values of 15-23 correspond to the nestling stage. The ge function will return a value of one (nestling stage) only when the statement is true.

Because the value of age for this nest is -3, the add function column returns a value of -3 (since 0 + -3 = -3) for the first row. The ge function (third column) returns a value of zero because the statement is false; age (-3) is not greater than or equal to 15. A value of one appears for the first time in row 19; here, the add function returns a value of 15 (since 18 + -3 = 15). The ge function returns a value of one because the statement is true; `add(18,age)` results in 15 which is greater than or equal to 15.

The fourth column produces an age slope variable that will be zero until the bird reaches 15 days of age, and then becomes equal to the bird's age. The result is that the age trend model of survival now changes to a different intercept and slope once the bird hatches.

Some useful tricks

An easy way to prepare these complicated sets of functions is to use Excel to prepare the values and then paste them into the design matrix. The following illustrates how to used the concatenate function in Excel to concatenate together a column and a closing ')' to create a complicated column of functions that duplicate the above example.

```
A   B                                C                                D

1   =concatenate("add(age,",A2,")")  =concatenate("GE(",B2,",15)")  =concatenate("product(",B2,",",C2,")")
2   =concatenate("add(age,",A3,")")  =concatenate("GE(",B3,",15)")  =concatenate("product(",B3,",",C3,")")
3   =concatenate("add(age,",A4,")")  =concatenate("GE(",B4,",15)")  =concatenate("product(",B4,",",C4,")")
...
```

Other Details

The design matrix values can have up to 60 characters, and unlimited nesting of functions (within the 60 character limit). As an example, the following is a very complicated way of computing a value of 1:

```
log(exp(log(exp(product(max(0,1),min(1,5))))))
```

Before the design matrix is submitted to the numerical optimizer, each entry in the design matrix is checked for a valid function name at the outermost level of nesting, plus that the number of '(' matches the number of ')'.

In previous versions of **MARK**, the design matrix functions were allowed to reference a value in one of the preceding columns. This capability was removed when the ability to nest functions was installed. No flexibility was lost with the removal of the 'Colxx' capability, and a considerable increase in versatility was obtained with the nested design matrix function calls. As shown in the Excel 'Tricks' example above, the ability to use values from other columns is still available. The 'Colxx' capability was also a very error prone method in that a column could be inserted ahead of the column being referenced, and the entire model would become nonsense without the user realizing that a mistake had been made. Therefore, the 'Colxx' capability was removed.

end sidebar

11.5. Visualizing the functional relationship with individual covariates

In the first example presented in this chapter, we considered the relationship between survival and individual body mass, under the hypothesis that there was strong 'normalizing selection' on mass - i.e., that the relationship between survival and mass was quadratic. We found that a quadratic model

$$\text{logit}(\hat{\varphi}) = 0.256733 + 1.1750545(\texttt{mass}_s) - 1.0555046(\texttt{mass}_s^2)$$

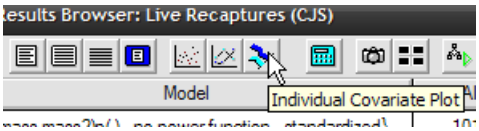
had good support in the data. We discussed briefly the mechanics of reconstituting the estimates of survival on the normal probability scale - the complication is that you need to generate a reconstituted value for each plausible value of the covariate(s) in the model. In fact, this is not particularly challenging for simple models such as this. Because the linear model consists of a covariate (mass) plus a function of the covariate ($\texttt{mass}^2$), it is relatively trivial to code this into a spreadsheet and generate a basic plot of predicted survival values over a range of values for mass. In fact, this is effectively what was done to generate the plot of predicted versus observed values we saw earlier (example on p. 14).

But, there are no confidence bounds on the predicted value function. The calculation of 95% CI for this function requires use of the *Delta method* - although not overly difficult to apply (the Delta method is discussed at length in Appendix B), it can be cumbersome and time consuming to program.

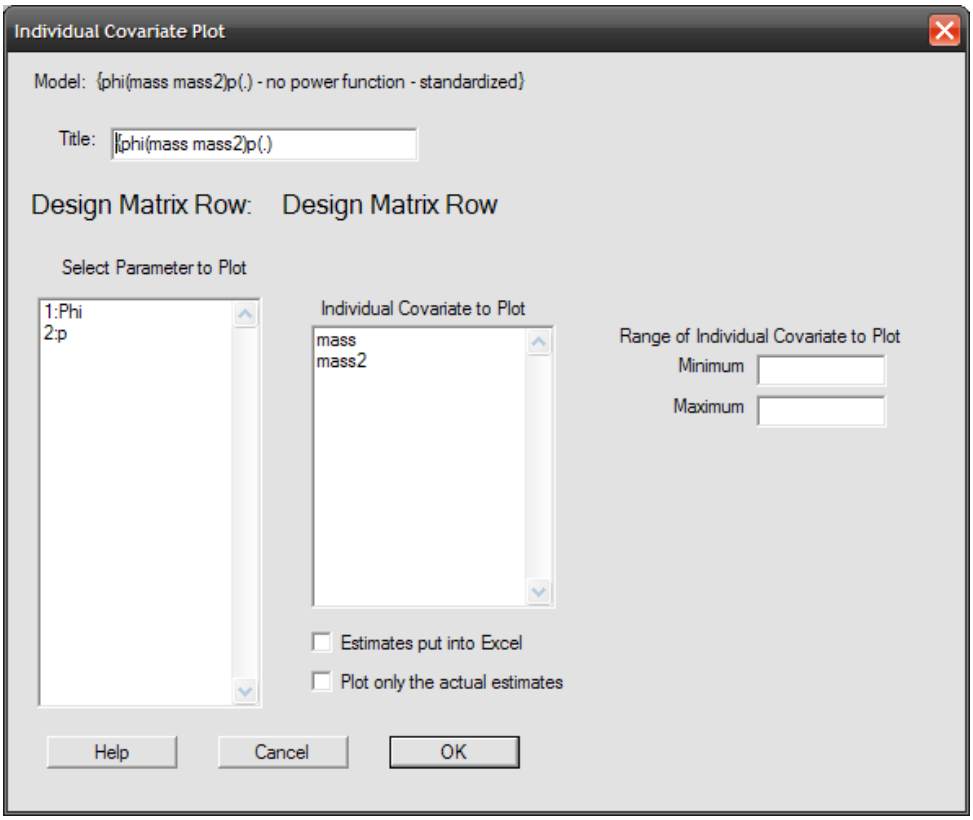
Fortunately, **MARK** has a plotting tool that make it convenient to generate a basic plot of predicted values from models with individual covariates, which includes the estimated 95% CI. For higher-resolution 'publication' plots, **MARK** makes it possible to output the data (including the data corresponding to the 95% CI) to a spreadsheet.

Let's demonstrate this for the analysis we previously completed on the normalizing selection data in `indcov1.inp`. Open up the `.DBF` file corresponding to those results, and retrieve the most parsimonious model from the model set we fit to those data $\{\varphi_{\texttt{mass mass}^2 p.}\}$.

Then, click on the '**`Individual Covariate Plot`**' icon in the main **MARK** toolbar:



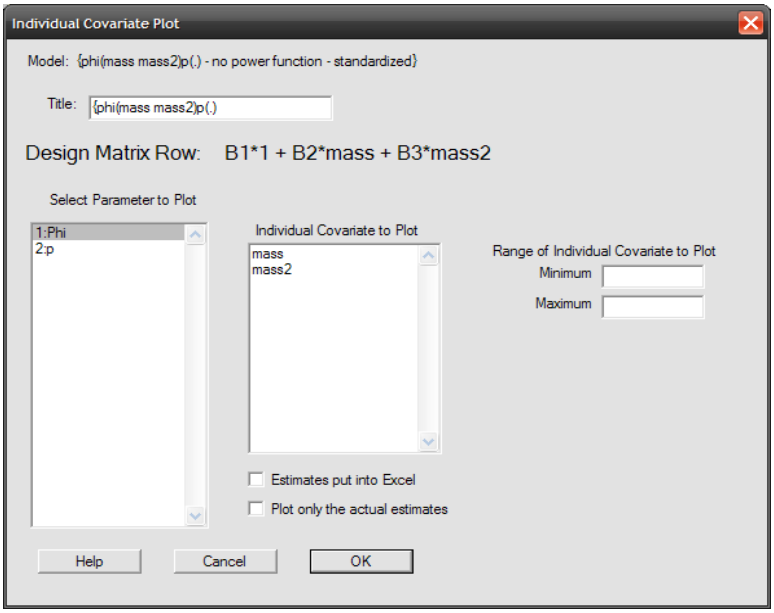
This will bring up a new window which will allow you to specify key attributes of the plot:



Notice that the title of the currently active model is already inserted in the title box. Next, are two boxes where you specify (i) which parameter you want to plot, and (ii) which individual covariate you want to plot. In our model, there are 2 different individual covariates - `mass` and `mass2`.

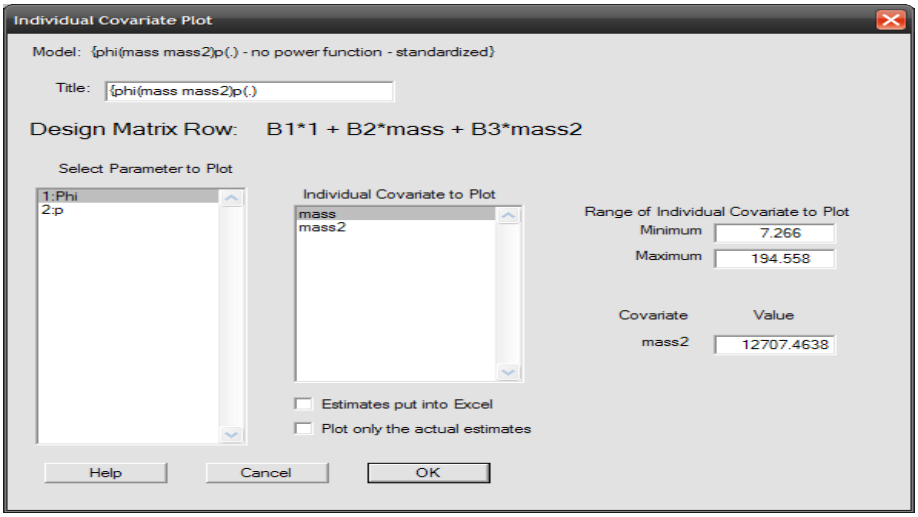
So, first question - which one to plot? If you look back at the figure at the bottom of p. 13, you'll see that we're interest in plotting 'survival' versus 'mass'. So, if our goal is to essentially replicate these plots, with the addition of 95% CI, using this individual covariate plot tool in **MARK**, it would seem to make sense that we should specify `mass` as the covariate we want to plot. Finally, two boxes which allow us to specify the numerical range of the individual covariate to plot. Also notice the check box you can check if you want to output the various estimates that go into the plot output to a spreadsheet.

OK – seems easy enough. Let's start by clicking on the survival parameter '**Phi**'.



As soon as we do so, the window 'updates', and now presents you with the '**Design Matrix Row**'. For this example, the DM has only 2 rows, so what is presented is in fact the linear model itself.

Next, we click on 'mass' to specify that as the individual covariate we want to plot. The window immediately updates - and spawns a new box in the process.

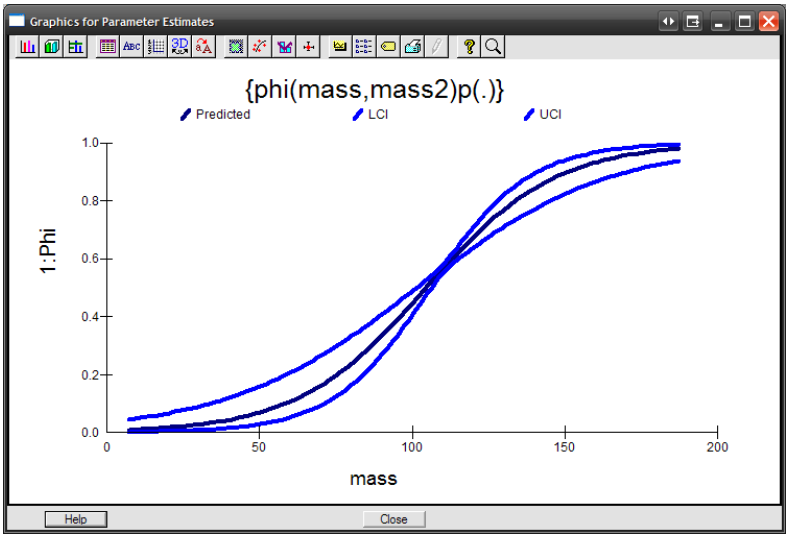


As you can see, the range of covariate values has been updated showing the maximum and minimum values that are actually in the .INP file. You can change these manually as you see fit (usual caveats about extrapolating a plot outside the range of the data apply).

Now, what about the new box - showing mass2 set to 12,707.4638? First, you might recognize the number - 12,707.4638 is the square of the mean mass of all individuals in the sample. But, why is a box

for mass2 there in the first place? It's there because the linear model that **MARK** is going to plot has 2 covariates - mass and mass2.

OK - so what does **MARK** actually plot? Well, if you click the '**OK**' button, **MARK** responds with



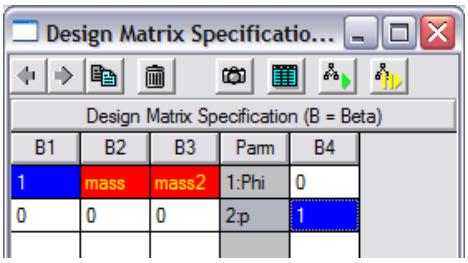
which doesn't look remotely like the quadratic curve we were expecting. What is actually being plotted? Well, if you think about it for a moment, it should be clear that **MARK** is plotting the functional relationship between survival and mass, holding the value of mass2 constant at the mean value! Different values of mass2 would yield different plots.

So, **MARK** isn't doing anything wrong - it's simply plotting what you told it to plot. **MARK** generates a 2-D plot between some parameter and one covariate. If there are other covariates in the model, then it needs to know what to do with them. Clearly, if there were only 2 covariates in the model, you could construct a 3-D plot (the two covariates on the x- and y-axes, and the parameter on the z-axis), but what if you had > 2 covariates? If would be difficult to program **MARK** to accommodate all permutations in the plot specification window, so it defaults to 2-D plots, meaning (i) you plot a parameter against only one covariate, and (ii) you need to tell **MARK** what to do with the other covariates.

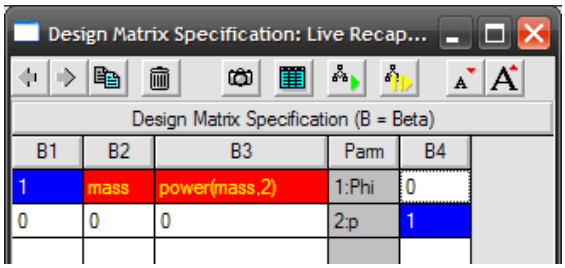
So, how do you tell **MARK** to plot survival versus mass and mass2 together, as a single 2-D plot? The key is in specifying the relationship between mass and mass2 explicitly - in effect, telling **MARK** that mass2 is in fact just (mass × mass). **MARK** doesn't 'know' that the second covariate (mass2) is a simple function of the first (mass). **MARK** doesn't know this because you haven't told **MARK** that this is the case. In your DM, you simply entered mass and mass2 as label names for the covariates, which were in fact 'hard-coded' in the .INP file. You (the user) know what they represent, but all **MARK** sees are two different covariates with two different labels.

So, if you can't pass this information to **MARK** in the plot specification window, where can you do so? *Hint*: what was the subject of the last -sidebar- presented several pages back? Looking back, you'll see that we introduced a series of 'design matrix functions', which included power and product. In our current analysis, we coded for mass and mass2 explicitly in the DM by entering the labels corresponding to the mass and mass2 covariates, which were hard-coded into the .INP file. As such, we know what the covariates represent, but **MARK** doesn't - it only knows the label names.

But, what if instead of

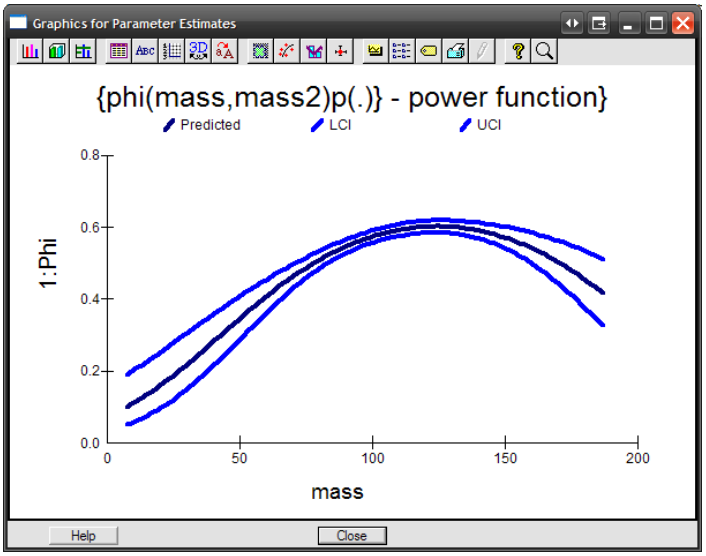


we used



Look closely at this second DM - notice that we've used the `power` function. Recall that the `power` function has two arguments - the first argument (`mass`, in this example) is raised to the power of the second argument (2, in this case). Now, we have explicitly coded (i.e., told **MARK**) that the second covariate is a power function of the first covariate. And because **MARK** now knows this, it knows what to plot, and how.

Run this model, and add the results to the browser. As expected, the results are identical to what we saw when we ran this model using the hard-coded `mass2` in the `INP` file. But, more importantly, when we plot this model, we get exactly what we were looking for:



Note that there are two other options in the '**Individual Covariate Plot**' specification window: you can (i) output the estimates into Excel, or (ii) plot only the actual estimates (meaning, plot only the reconstituted estimates for the parameter for the actual covariates in the input file - the estimate are presented without their estimated SE).

Beyond the mechanics of plotting individual covariate functions, which is clearly part of the intent of this section, this example also demonstrates one of the 'hidden' advantages of using the DM functions to handle coding any functional relationships you might have among your covariates. Not only does this save you from having to do those calculations by hand while you construct the `INP` file, they also provide a convenient mechanism to make those functional relationships 'known' to **MARK**.

Final word - at present, the individual covariate plot function in **MARK** is restricted to single models only - it cannot plot model averaged functions with individual covariates (see section 11.8). However, plotting model averaged models with covariates *is* possible using **RMark** (see Appendix C - discussion of the `covariate.predictions` function).

begin sidebar

AIC, BIC - example of the difference

Back in Chapter 4, we briefly introduced two different information theoretic criteria which can be used to assist in model selection, the AIC (which we've made primary use of), and the BIC. Recall that we briefly discussed the differences between the two - noting that (in broad, simplified terms), the AIC has a tendency to pick overly complex models - especially if the 'true' model structure is complex, whereas the BIC has a tendency to pick overly simple models when the reverse is true.

We can demonstrate these differences by contrasting the results of model selection using AIC or BIC for our analysis of the normalizing selection data. To highlight differences between the two, we'll consider the following 4 models: $\{\varphi_{.}p_{.}\}$, $\{\varphi_{(mass)}p_{.}\}$, $\{\varphi_{(mass,mass^2)}p_{.}\}$, and $\varphi_{(mass,mass^2,mass^3)}p_{.}\}$. Recall that the *true* model used to generate the simulated data was model $\{\varphi_{(mass,mass^2)}p_{.}\}$. So, our candidate model set consists of two models which are simpler than the 'true' model, and one model that is more complex than the 'true' model.

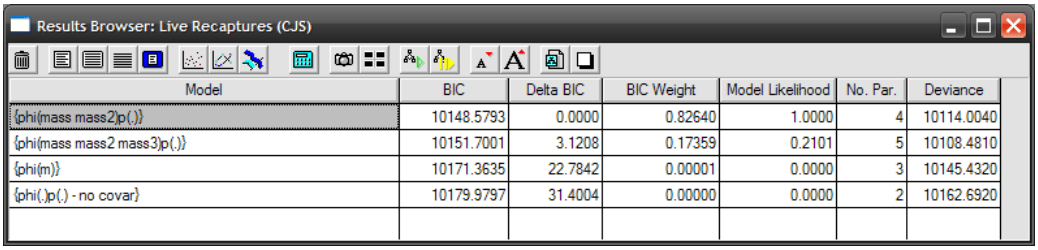
Here are the results from fitting the candidate model set to the data, using AIC as the model selection criterion:

Results Browser: Live Recaptures (CJS)

Model	AICc	Delta AICc	AICc Weight	Model Likelihood	No. Par.	Deviance
{phi(mass mass2 mass3)p(.)}	10118.4916	0.0000	0.85318	1.0000	5	10108.4810
{phi(mass mass2)p(.)}	10122.0111	3.5195	0.14682	0.1721	4	10114.0040
{phi(m)}	10151.4362	32.9446	0.00000	0.0000	3	10145.4320
{phi(.)p(.) - no covar}	10166.6941	48.2025	0.00000	0.0000	2	10162.6920

Note that although model $\{\varphi_{(mass,mass^2)}p_{.}\}$ is the true generating model, it was not the most parsimonious model using AIC - in fact, it was 5-6 times less well supported than was a more complex model $\{\varphi_{(mass,mass^2,mass^3)}p_{.}\}$.

What happens if we use BIC as our model selection criterion? (Remember this can be accomplished by changing **MARK**'s preferences; '**File | Preferences**'). If you look at the results browser at the top of the next page, you'll see that the BIC selected what we know to be the 'true' model $\{\varphi_{(mass,mass^2)}p_{.}\}$ - the next best model $\{\varphi_{(mass,mass^2,mass^3)}p_{.}\}$ was 5-6 times less well supported than was the most parsimonious model.


Results Browser: Live Recaptures (CJS)

Model	BIC	Delta BIC	BIC Weight	Model Likelihood	No. Par.	Deviance
{phi(mass mass2)p(.)}	10148.5793	0.0000	0.82640	1.0000	4	10114.0040
{phi(mass mass2 mass3)p(.)}	10151.7001	3.1208	0.17359	0.2101	5	10108.4810
{phi(m)}	10171.3635	22.7842	0.00001	0.0000	3	10145.4320
{phi(.)p(.) - no covar}	10179.9797	31.4004	0.00000	0.0000	2	10162.6920

So, is this an example of BIC 'doing better' when the true model is relatively simple? Or is the fact that the BIC picked the right model an artifact of the inclusion of the right model in the candidate model set (a point of some contention in the larger discussion)? Our point here is not to make conclusions one way or the other. Rather, it is merely to demonstrate the fact that different model selection criterion can yield quite different results (conclusions) - so much so (at least on occasion) that it will be worth you spending some time thinking hard about the general question, and reading the pertinent literature. As mentioned in Chapter 4, good starting points are

- Burnham & Anderson (2004) Multimodel inference - understanding AIC and BIC in model selection. *Sociological Methods & Research*, **33**, 261-304.
- Link & Barker (2006) Model weights and the foundations of multimodel inference. *Ecology*, **87**, 2626-2635

end sidebar

11.6. Missing covariate values, time-varying covariates, and other complications...

Several strategies for handling missing individual covariates are available. Probably the best option is to code missing individual covariate values with the mean of the variable for the population measured. Replacing the missing value with the average means that the mean of the observed values will not change, although the variance will be slightly smaller because all missing values will be exactly equal to the mean and hence not variable. The easiest way to accomplish this in **MARK** is to use the 'standardize covariates' option - if you compute the mean of the non-missing values of an individual covariate, and then scale the non-missing values to have a mean of zero, the missing values can be included in the analysis as zero values, and will not affect the value of the estimated β term. (*note*: we don't advise this trick for a covariate with a large percentage of missing values because you have no power, but this approach does work for a 'small' number of missing values).

If you have lots of missing values, another option is to code the animals into 2 groups, where all the missing values are in one group. Then, you can use both groups to estimate a common parameter, and only apply the individual covariate to one group. This approach can be tricky, so think through what you are doing before you try it.

What about covariates that vary through time? In all our examples so far, we've made the assumption that the covariate is a constant over the lifetime of the animal. But, clearly, this will often (perhaps generally) not be the case. For example, consider body mass. Body mass typically changes dynamically over time, and if we believe that body mass influences survival or some other parameter, then we might want to constrain our estimates to be functions of a dynamically changing covariate, rather than a static one (typically measured at the time the individual was initially captured and marked). You can handle time-varying covariates in one of a couple of ways.

First, you **can** include time-varying individual covariates in **MARK** files, but you must have a value for every animal on every occasion, even if the animal is not captured. Typically, you can impute these values if they are missing (not observed), but be sure to recognize what this imputation might do to your estimates. You implement time-varying individual covariates just like any other individual covariate, except that you have to have a different name for each covariate corresponding to each time period. For example, suppose you have a known fate model with 5 occasions, and you have estimated the parasite load for each animal at the beginning of each of the 5 occasions. The 5 values for each animal are contained in the variables `var1`, `var2`, `var3`, `var4`, and `var5`. A design matrix that would estimate the effect of the parasite load assuming that the effect is constant across time would be:

```
1 var1 1 var2 1 var3 1 var4 1 var5
```

The second β estimate is the slope parameter associated with the time-varying individual covariates. Note that you do not want to standardize these individual covariates, because standardizing them will cause them to no longer relate to one another on the same scale (making a common slope parameter nonsensical). Each would have a different scale after standardizing. If you need to standardize the covariates, you must do so before the values are included in a **MARK** encounter histories input file, and you must use a common mean and standard deviation across the entire set of variables and observations. For further details on this approach, consult the **MARK** help file.

Alternatively, you can 'discretize' the covariate, and use a multi-state model (chapter 10) to model transitions as a function of the covariate 'class' the individual is in. For example, suppose you believe that survival from time (i) to (i+1) is strongly influenced by the size of the organism at time (i). Now, size is clearly a continuously distributed trait. But, perhaps you might reasonably classify each marked individual as either 'large', 'average', or 'small' size. Then, each individual at each occasion is classified into one of these 3 different size classes, and you use a multi-state approach to estimate the probability of surviving as a function of being in a particular size class. If the covariate is not measured (typically, if the individual is not captured), then the missing value is accounted for explicitly by including the encounter probability p in the model. Moreover, you would also be able to look at the relationship between survival as a function of size, and the probability of moving among size classes. Sounds reasonable, but you need to consider a couple of things. First, in applying this approach, you are discretizing a continuous distribution, and how many discrete categories you use, and how you decide to partition them (e.g., what criterion you use to define a 'large' versus 'small' individual), may strongly influence the results you get. However, when there are a large number of missing covariate values, or if discretizing seems 'reasonable', this is a fairly robust and easily implemented approach. Second, you might need to be a bit 'clever' in setting up your design matrix to account for trends (relationships) among states, as we'll see in the following worked example.

11.6.1. Continuous individual covariates & multi-state models: a worked example

Let's work through an example - not only to demonstrate an application of multi-state modeling to this sort of problem (giving you a chance to practice what you learned in Chapter 10), but also to force you to think deeply (yet again) about the building of a design matrix.

Consider a situation where we believe there is strong *directional* selection on (say) body size, where larger individuals have higher survival than do smaller individuals. Suppose we have categorized individuals as 'small' (**S**), 'medium' (**M**) and 'large' (**L**).

We simulated a 6-occasion data set (`ms_directional.inp`) according to the following parameter values for 'size-specific survival':

	interval				
state	1	2	3	4	5
S	0.500	0.700	0.600	0.700	0.700
M	0.525	0.749	0.624	0.749	0.749
L	0.551	0.801	0.649	0.801	0.801

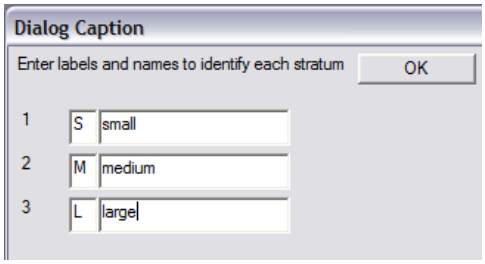
If you look closely, you'll see that within each interval, the difference in the latent survival probability used in the simulation differs by a constant multiplicative factor such that there is a linear increase in survival with size. However, if you look even more closely, you'll note that the rate of this increase in survival with size is not constant over intervals. So, imagine that for each time interval, you calculate the slope of the relationship between survival and size. This slope should show heterogeneity among intervals (i.e., the strength of directional selection on size varies over time).

To make things 'fun' (i.e., more realistic) we'll also specify some size-specific transition parameters:

		from		
		S	**M**	**L**
	S	0.7	0.0	0.0
to	**M**	0.2	0.8	0.0
	L	0.1	0.2	1.0

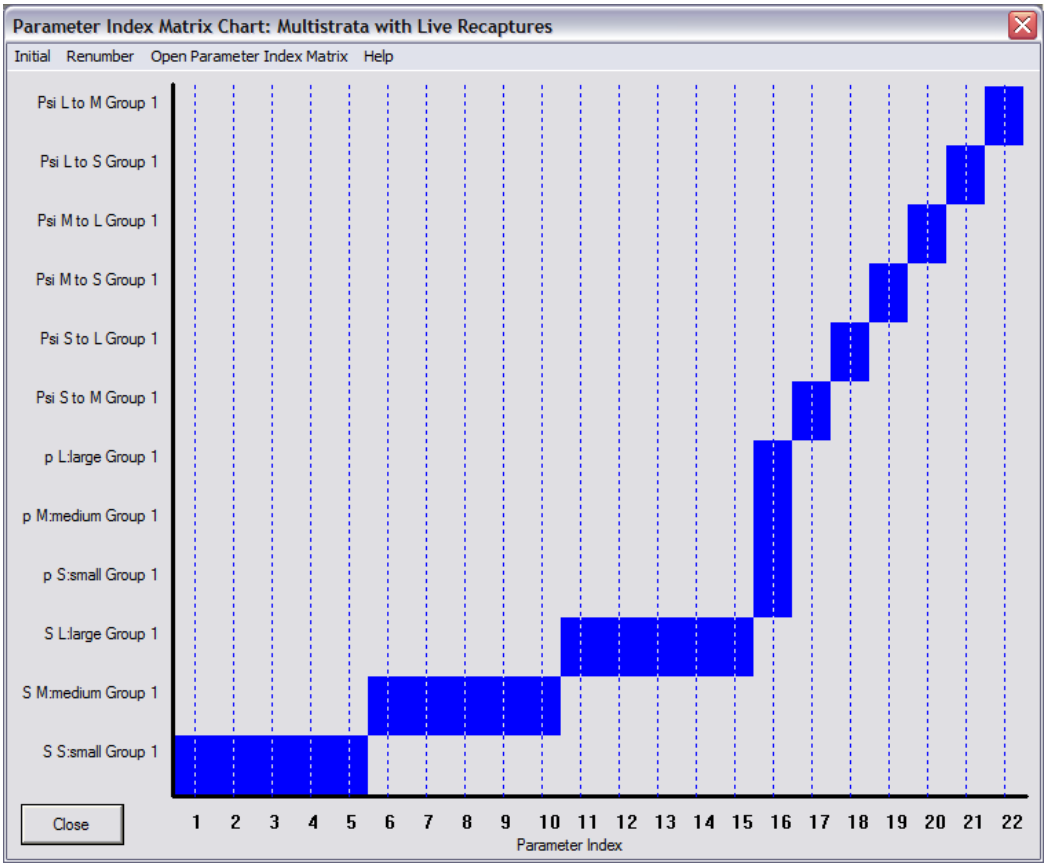
So, small (**S**) and medium (**M**) individuals can stay in the same size class or grow over a given interval, but individuals cannot get smaller. We'll assume that the encounter probability for all size classes and all intervals is the same; however, to make this even more realistic, we'll assume that $p = 0.7$ for all size classes - since $p < 1$, then we have 'missing covariates'.

So, start **MARK**, and begin a new '**`multi-state`**' analysis: select the `ms_directional.inp` file, and specify 6 occasions, and 3 states:

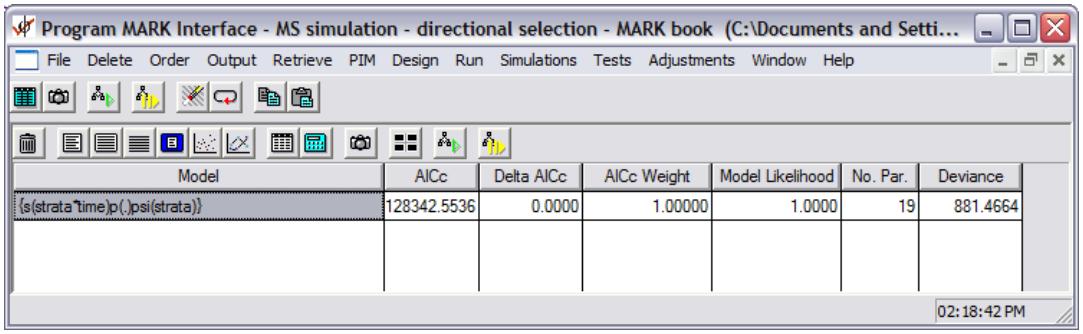


We'll start with a general model with time-dependence in survival, among states, and among time intervals. We'll make the encounter parameter p constant among states and over time, and will make ψ constant within state.

This general structure is reflected in the following PIM chart:



Now, before we run this model, we have to consider if there are any parameters we need to fix (due to logical constraints). As noted earlier, some of the transitions are not possible; specifically, $\psi^{MS} = \psi^{LM} = \psi^{LS} = 0$. Thus, looking at our PIM chart, we see that this corresponds to setting parameters 19, 21 and 22 to 0. Go ahead and fix these parameters in the numerical estimation setup window. Call this model '`s(state*time)p(.)psi(state)`', run it, and add the results to the browser.



If we look at the estimates, we'll see that, by and large, the values are consistent with the underlying model structure.

OK - on to the 'clever' design matrix we alluded to before. The model we just fit is a naïve model, as far as our underlying hypothesis is concerned – it is a model which simply allows the estimates for survival to vary among states, and over time. In essence, a simple heterogeneity model. By itself, this is not particularly interesting, although it is arguably a reasonable null model.

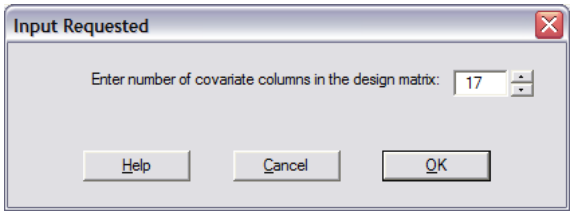
But, we're interested in a particular *a priori* hypothesis: specifically, that survival increases with size. We may also suspect that the strength (magnitude) of this directional selection favoring larger sized individuals varies over time. So, what we want to fit is a model where, within a given interval, survival is constrained to be a linear function of size (i.e., follow a trend), and that the slope of this trend may vary over time.

So, here's the tricky bit – in effect, we're now going to treat each time interval as a group, and ask if the slope of a relationship between survival and size varies among levels of this group (i.e., among time intervals). So, we need to figure out how to do two things: (1) build a design matrix where each time interval is a group, and (2) within a time interval, have survival constrained to follow a trend with size among states (i.e., an ordinal constraint on survival with increasing size). How do we do this?

Well, with a bit of thought, you might see your way to the solution. First, start by writing out the linear model. We know we need an intercept (β_1). There are 6 occasions, so 5 time intervals, meaning we need 4 columns in the design matrix to code for the `TIME` grouping ($\beta_2 \rightarrow \beta_5$). Next, we want to impose a `TREND` over states. Recall from Chapter 6 how we handled trends: a single column consisting of an ordinal series. So, for `TREND`, one column (β_6). Next, the interaction term of `TIME` and `TREND` - $(4 \times 1) = 4$ columns for the interaction terms ($\beta_7 \rightarrow \beta_{10}$). So, for the survival parameter,

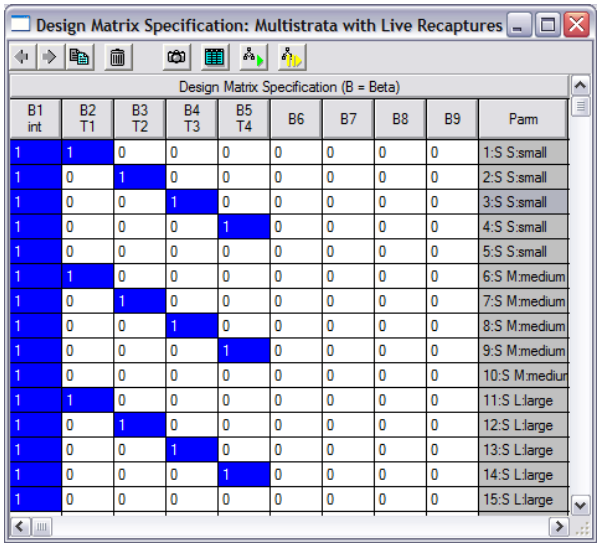
$$\begin{aligned} S = {} & \beta_1 \\ & + \beta_2(\texttt{T1}) + \beta_3(\texttt{T2}) + \beta_4(\texttt{T3}) + \beta_5(\texttt{T4}) \\ & + \beta_6(\texttt{TREND}) \\ & + \beta_7(\texttt{T1.TREND}) + \beta_8(\texttt{T2.TREND}) + \beta_9(\texttt{T3.TREND}) + \beta_{10}(\texttt{T4.TREND}) \end{aligned}$$

Now, encounter probability p is constant among states and over time, so one column (β_{11}) for that parameter. For the ψ parameters, one for each of the estimated transitions. Remember that if there are n states that there are $n(n-1)$ estimated transitions, then for 3 size states, $3(3-1) = 6$ transitions, meaning 6 columns ($\beta_{12} \rightarrow \beta_{17}$). So, in total, our design matrix should have 17 columns. So, we tell **MARK** we want to build a reduced design matrix, with 17 columns:



MARK will then respond by giving us a 'blank' design matrix with 17 columns. Now, we need to put the appropriate dummy variable structure into those columns.

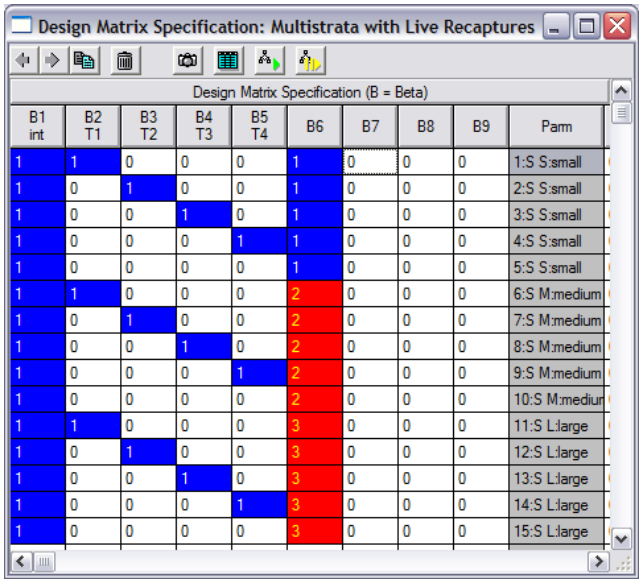
OK, starting our design matrix is easy enough: a column of 15 '1's for the intercept. Then, looking back at our linear model, we see that we next want to code for the 5 TIME intervals: 4 columns ($\beta_1 \rightarrow \beta_4$). We use the same coding scheme we're familiar with - all we want to do is make sure the dummy-variable structure unambiguously indicates the time interval:



B1 int	B2 T1	B3 T2	B4 T3	B5 T4	B6	B7	B8	B9	Parm
1	1	0	0	0	0	0	0	0	1:S S:small
1	0	1	0	0	0	0	0	0	2:S S:small
1	0	0	1	0	0	0	0	0	3:S S:small
1	0	0	0	1	0	0	0	0	4:S S:small
1	0	0	0	0	0	0	0	0	5:S S:small
1	1	0	0	0	0	0	0	0	6:S M:medium
1	0	1	0	0	0	0	0	0	7:S M:medium
1	0	0	1	0	0	0	0	0	8:S M:medium
1	0	0	0	1	0	0	0	0	9:S M:medium
1	0	0	0	0	0	0	0	0	10:S M:mediur
1	1	0	0	0	0	0	0	0	11:S L:large
1	0	1	0	0	0	0	0	0	12:S L:large
1	0	0	1	0	0	0	0	0	13:S L:large
1	0	0	0	1	0	0	0	0	14:S L:large
1	0	0	0	0	0	0	0	0	15:S L:large

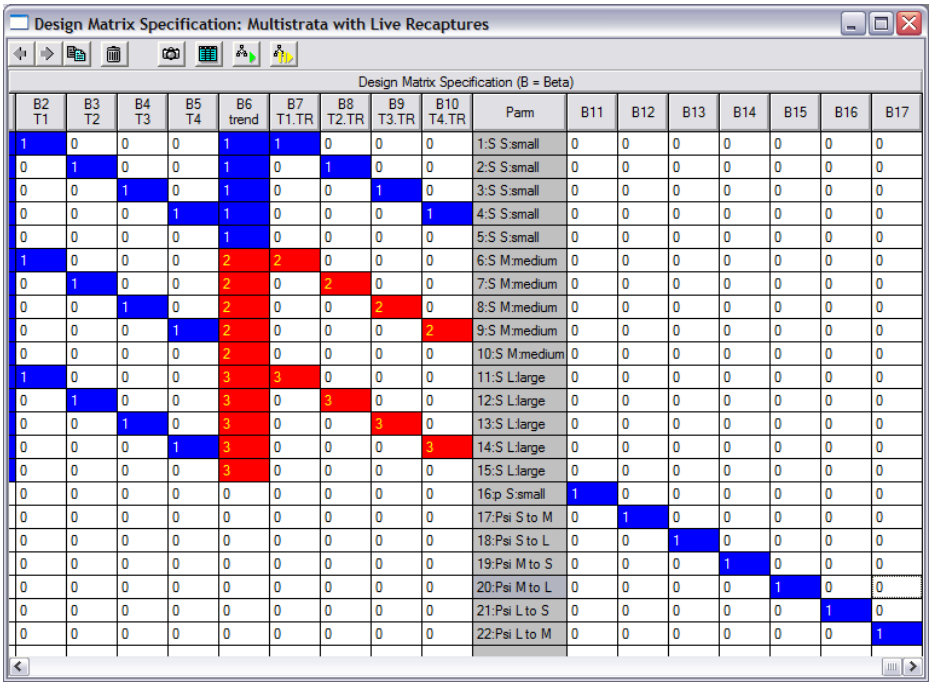
So far, so good. Now, for the 'hard part'. We now need to code for TREND. But, remember, here, we're not coding for TREND over TIME, but rather, TREND over states *within* TIME. You might remember that if we have 3 levels we want to constrain some estimate to follow a trend over, then we can use the ordinal sequence 1, 2, and 3 as the TREND covariate (check the relevant sections of Chapter 6 if you're unsure here). But, where do we put these TREND coding variables?

The key is remembering – TREND *among* states *within* TIME interval. So, here is how we code TREND for this model:



B1 int	B2 T1	B3 T2	B4 T3	B5 T4	B6	B7	B8	B9	Parm
1	1	0	0	0	1	0	0	0	1:S S:small
1	0	1	0	0	1	0	0	0	2:S S:small
1	0	0	1	0	1	0	0	0	3:S S:small
1	0	0	0	1	1	0	0	0	4:S S:small
1	0	0	0	0	1	0	0	0	5:S S:small
1	1	0	0	0	2	0	0	0	6:S M:medium
1	0	1	0	0	2	0	0	0	7:S M:medium
1	0	0	1	0	2	0	0	0	8:S M:medium
1	0	0	0	1	2	0	0	0	9:S M:medium
1	0	0	0	0	2	0	0	0	10:S M:mediur
1	1	0	0	0	3	0	0	0	11:S L:large
1	0	1	0	0	3	0	0	0	12:S L:large
1	0	0	1	0	3	0	0	0	13:S L:large
1	0	0	0	1	3	0	0	0	14:S L:large
1	0	0	0	0	3	0	0	0	15:S L:large

Holy smokes! OK, after you've caught your breath (or had a beer or two), it's actually not that bad. Remember, `TREND` *among* states *within* `TIME` interval. So, for the first interval for the 3 states, corresponding to rows 1, 6, and 11, respectively, we enter 1, 2 and 3. Similarly, for the second interval for the 3 states, corresponding to rows 2, 7, and 12, respectively, we again enter 1, 2 and 3, and so on for each of the intervals. Think about this - remember, `TIME` *is a grouping variable for this model*. After all this, the interaction terms (and the encounter and transition parameters) are straightforward (the full design matrix is shown below).

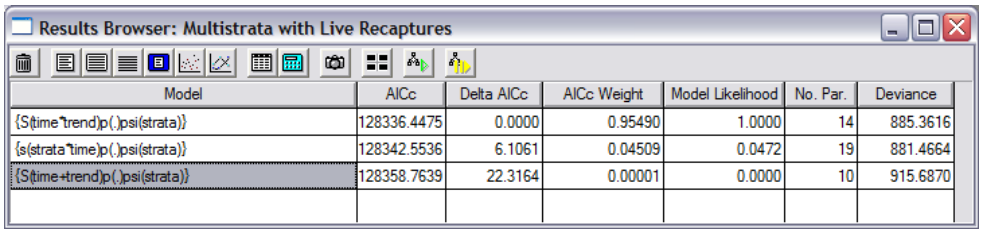


Design Matrix Specification: Multistrata with Live Recaptures

Design Matrix Specification (B = Beta)

B2 T1	B3 T2	B4 T3	B5 T4	B6 trend	B7 T1.TR	B8 T2.TR	B9 T3.TR	B10 T4.TR	Parm	B11	B12	B13	B14	B15	B16	B17
1	0	0	0	1	1	0	0	0	1:S S:small	0	0	0	0	0	0	0
0	1	0	0	1	0	1	0	0	2:S S:small	0	0	0	0	0	0	0
0	0	1	0	1	0	0	1	0	3:S S:small	0	0	0	0	0	0	0
0	0	0	1	1	0	0	0	1	4:S S:small	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0	5:S S:small	0	0	0	0	0	0	0
1	0	0	0	2	2	0	0	0	6:S M:medium	0	0	0	0	0	0	0
0	1	0	0	2	0	2	0	0	7:S M:medium	0	0	0	0	0	0	0
0	0	1	0	2	0	0	2	0	8:S M:medium	0	0	0	0	0	0	0
0	0	0	1	2	0	0	0	2	9:S M:medium	0	0	0	0	0	0	0
0	0	0	0	2	0	0	0	0	10:S M:medium	0	0	0	0	0	0	0
1	0	0	0	3	3	0	0	0	11:S L:large	0	0	0	0	0	0	0
0	1	0	0	3	0	3	0	0	12:S L:large	0	0	0	0	0	0	0
0	0	1	0	3	0	0	3	0	13:S L:large	0	0	0	0	0	0	0
0	0	0	1	3	0	0	0	3	14:S L:large	0	0	0	0	0	0	0
0	0	0	0	3	0	0	0	0	15:S L:large	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	16:p S:small	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	17:Psi S to M	0	1	0	0	0	0	0
0	0	0	0	0	0	0	0	0	18:Psi S to L	0	0	1	0	0	0	0
0	0	0	0	0	0	0	0	0	19:Psi M to S	0	0	0	1	0	0	0
0	0	0	0	0	0	0	0	0	20:Psi M to L	0	0	0	0	1	0	0
0	0	0	0	0	0	0	0	0	21:Psi L to S	0	0	0	0	0	1	0
0	0	0	0	0	0	0	0	0	22:Psi L to M	0	0	0	0	0	0	1

Go ahead and run this model - call it '`s(time*trend)p(.)psi(state)`', where the `time*trend` part indicates an interaction of the trend among `TIME` intervals. Run the model, and add the results to the browser. Then, build the additive model (by deleting the interaction columns from the design matrix) - call this model '`s(time+trend)p(.)psi(state)`', and again, add the results to the browser.



Results Browser: Multistrata with Live Recaptures

Model	AICc	Delta AICc	AICc Weight	Model Likelihood	No. Par.	Deviance
{S(time*trend)p(.)psi(strata)}	128336.4475	0.0000	0.95490	1.0000	14	885.3616
{s(strata*time)p(.)psi(strata)}	128342.5536	6.1061	0.04509	0.0472	19	881.4664
{S(time+trend)p(.)psi(strata)}	128358.7639	22.3164	0.00001	0.0000	10	915.6870

We see clearly that our model constraining survival to show a trend among states, with full interaction among time intervals, is by far the best supported model. Of course, this isn't surprising, since the data were simulated under this model. So - a fairly long stroll through an example of using a multi-state approach to handle covariates which vary through time. And, yet another example of why it is important to have a significant level of comfort with design matrices - unless you do, you won't be able to build the 'fancy models' you'd like to.

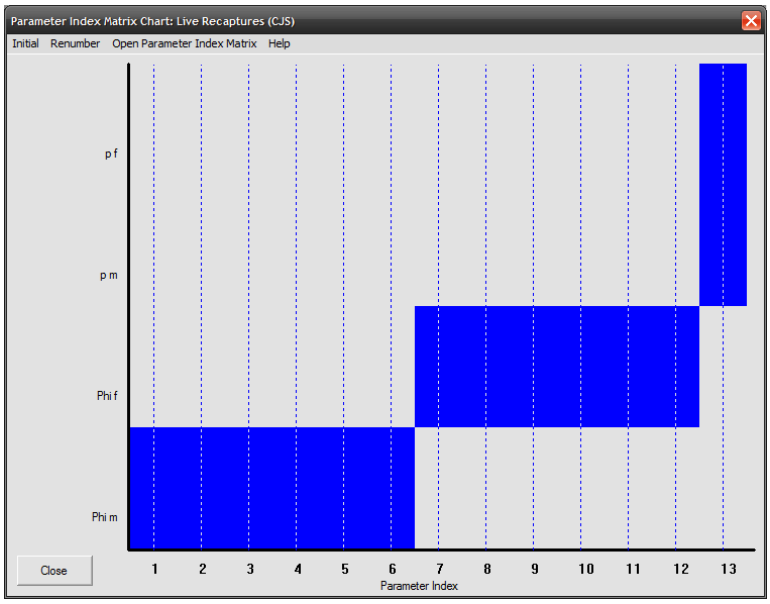
11.7. Individual covariates as 'group' variables: a useful short-cut?

Suppose you were interested in whether or not survival probability differed between male and female dippers. Having come this far in the book, you'll probably regard this as a trivial exercise - you specify the PIMs corresponding to the two sexes, perhaps construct the corresponding design matrix, and proceed to fit the models in your candidate model set. This is all fairly straightforward, and easy to implement - in part because the problem is sufficiently 'small' (meaning, only two parameters, relatively few occasions, only two groups) that the overall number, and complexity of the PIMs you construct (and the corresponding design matrix) is small. But, as we've seen, especially for 'large' problems (many parameters, many occasions, many PIMs), manipulating all the PIMs and the design matrix can become cumbersome (even given the convenience of manipulating the PIMs using the PIM chart).

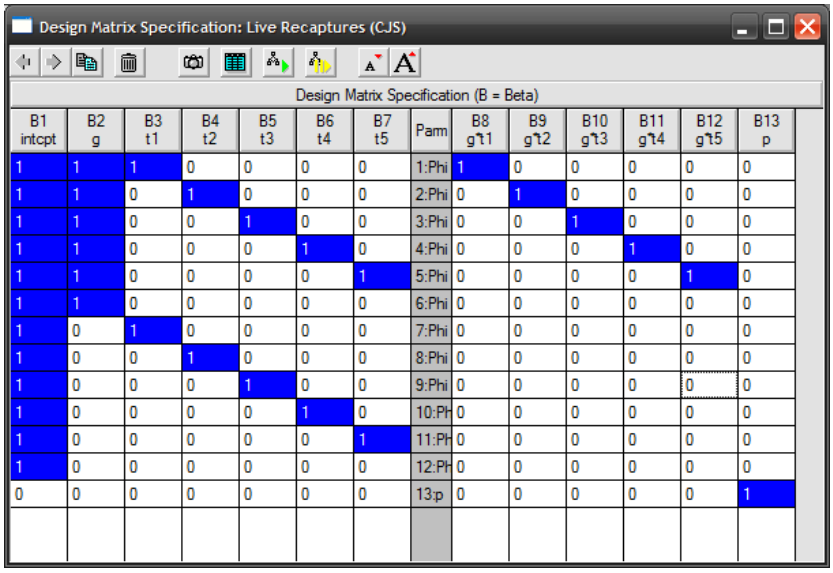
Is there an option? Well, as you might guess, given that this chapter concerns the use of individual covariates, you can, for a number of categorical models, use an alternative approach based on individual covariates. Such an approach can in some cases be easier and more efficient to implement. We'll consider a couple of examples here, starting with the dippers.

11.7.1. Individual covariates for a binary classification variable

Let's consider fitting the following 3 candidate models to the data collected for male and female dippers: $\{\varphi_{g*t}p_{.}\}$, $\{\varphi_{g+t}p_{.}\}$, $\{\varphi_{g}p_{.}\}$, where g is the 'grouping' variable - in this case, sex (male or female). Recall that the dipper data (`dipper.inp`) consist of live encounter data collected over 7 encounter occasions. We specify 2 attribute groups in the data type specification window in **MARK** (which we'll label `m` and `f`, respectively), and proceed to fit the three models in the candidate model set. To specify the underlying parameter structure for our general model $\{\varphi_{g*t}p_{.}\}$, we'll use fully time-dependent PIMs for survival, and constant PIMs for the encounter rate. The PIM chart looks like



and the corresponding design matrix is

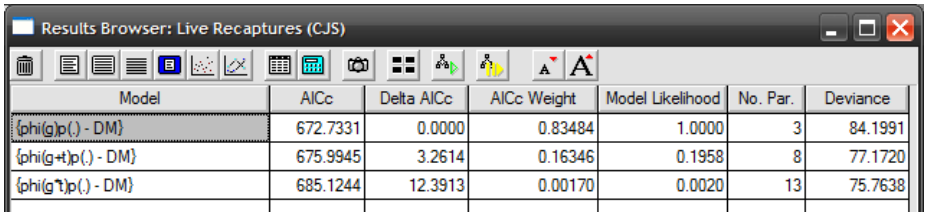


Design Matrix Specification: Live Recaptures (CJS)

Design Matrix Specification (B = Beta)

B1 intcpt	B2 g	B3 t1	B4 t2	B5 t3	B6 t4	B7 t5	Parm	B8 gt1	B9 gt2	B10 gt3	B11 gt4	B12 gt5	B13 p
1	1	1	0	0	0	0	1:Phi	1	0	0	0	0	0
1	1	0	1	0	0	0	2:Phi	0	1	0	0	0	0
1	1	0	0	1	0	0	3:Phi	0	0	1	0	0	0
1	1	0	0	0	1	0	4:Phi	0	0	0	1	0	0
1	1	0	0	0	0	1	5:Phi	0	0	0	0	1	0
1	1	0	0	0	0	0	6:Phi	0	0	0	0	0	0
1	0	1	0	0	0	0	7:Phi	0	0	0	0	0	0
1	0	0	1	0	0	0	8:Phi	0	0	0	0	0	0
1	0	0	0	1	0	0	9:Phi	0	0	0	0	0	0
1	0	0	0	0	1	0	10:Ph	0	0	0	0	0	0
1	0	0	0	0	0	1	11:Ph	0	0	0	0	0	0
1	0	0	0	0	0	0	12:Ph	0	0	0	0	0	0
0	0	0	0	0	0	0	13:p	0	0	0	0	0	1

We'll skip the details on how to modify this design matrix to specify the remaining two models in the model set (you should be pretty familiar with this by now). The results of fitting the three models to the dipper data are shown below:



Results Browser: Live Recaptures (CJS)

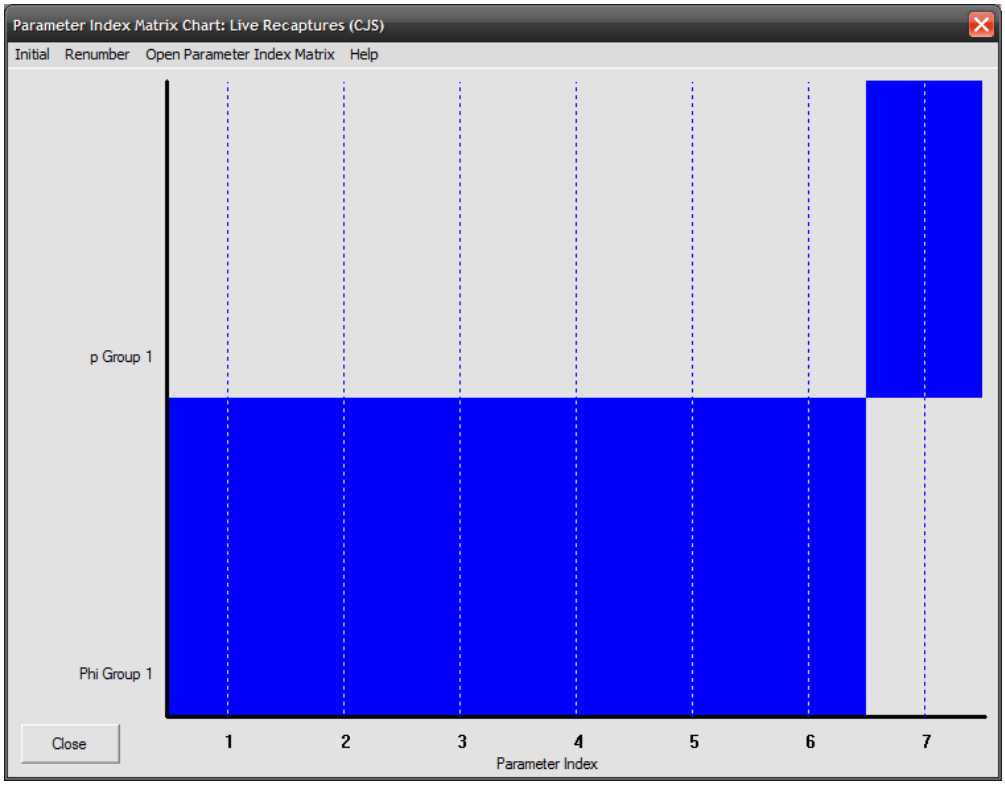
Model	AICc	Delta AICc	AICc Weight	Model Likelihood	No. Par.	Deviance
{phi(g)p(.) - DM}	672.7331	0.0000	0.83484	1.0000	3	84.1991
{phi(g+t)p(.) - DM}	675.9945	3.2614	0.16346	0.1958	8	77.1720
{phi(gt)p(.) - DM}	685.1244	12.3913	0.00170	0.0020	13	75.7638

Now, let's consider using an individual covariate approach to fitting the same three models to the dipper data. Our first step involves reformatting the input file. We need to reformat the input file to specify gender as an individual covariate. Much like with the design matrix, you need to consider how many covariates you need to specify group (in this case, sex). Clearly, in this case, the grouping variable is binary (has only two states), and thus we need only a single covariate to indicate group (sex). How do we reformat our data, using a single covariate to indicate sex? We'll use '1' to indicate males, and '0' to indicate females. Now, we reformat the dipper data as follows - consider the following table of different encounter histories (selected from the original `dipper.inp` file in 'standard' format), which we've transformed to use an individual covariate approach:

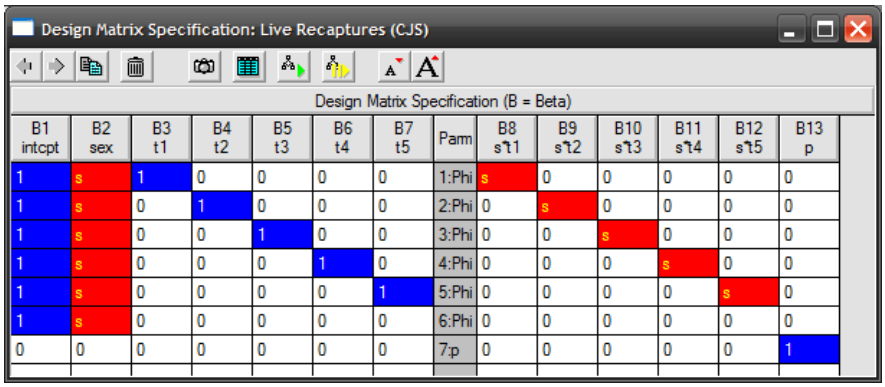
standard	*reformatted*
`1111110 1 0;`	`1111110 1 1;`
`1111100 0 1;`	`1111100 1 0;`
`1111000 1 0;`	`1111000 1 1;`
`1111000 0 1;`	`1111000 1 0;`
`1101110 1 0;`	`1101110 1 1;`

The key is to remember that under the original 'standard' formatting, there is one column in the input file for each of the groups: two sexes, two columns following the encounter history itself. So, a '1 0' indicates male (1 in the male column, 0 in the female column), and a '0 1' indicates a female (0 in the male column, 1 in the female column). When using an individual covariates approach, you have only one column for the covariate. But, notice there are 2 columns after the encounter history. Why? Don't we need just 1 covariate column? Yes, but remember that we also need a column of '1's' to indicate the frequency of number of individuals with a given encounter history (and since we're working with individual covariates, each encounter history corresponds to one individual, hence the frequency column has a '1' in it for each individual history). The first column after the encounter history is the frequency, and the second column is the covariate column for group (sex). So, a male in the original file (indicated by '1 0') becomes '1 1' in the reformatted file, and a female in the original file (indicated by '0 1') becomes '1 0' in the reformatted file. The reformatted data are contained in the file dipper_ind.inp (we'll leave it to you to figure out an efficient way to transform your data from one format to the other).

Now, when we specify the data type in **MARK**, we do not indicate 2 attribute groups, but instead change the default number of individual covariates from 0 to 1. We'll call this covariate s (for sex). If we make the encounter probability constant, the corresponding PIM chart should look like the one pictured at the top of the next page. Note that there are now only 6 parameters in the PIM chart for survival, instead of the 12 parameters specified in the PIM chart of our general model using the standard input format. Obviously, we're going to need to make up the difference somehow. In fact, you may have already guessed - by entering the individual covariates into the design matrix.



For our general model $\{\varphi_{g*t}p_{.}\}$, here is the corresponding design matrix using individual covariates:



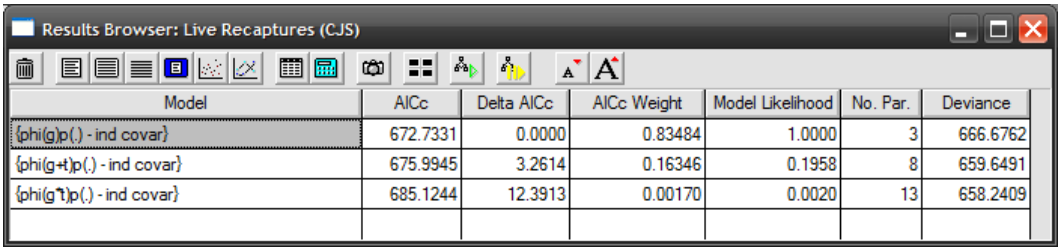
Design Matrix Specification: Live Recaptures (CJS)

Design Matrix Specification (B = Beta)

B1 intcpt	B2 sex	B3 t1	B4 t2	B5 t3	B6 t4	B7 t5	Parm	B8 s*t1	B9 s*t2	B10 s*t3	B11 s*t4	B12 s*t5	B13 p
1	s	1	0	0	0	0	1:Phi	s	0	0	0	0	0
1	s	0	1	0	0	0	2:Phi	0	s	0	0	0	0
1	s	0	0	1	0	0	3:Phi	0	0	s	0	0	0
1	s	0	0	0	1	0	4:Phi	0	0	0	s	0	0
1	s	0	0	0	0	1	5:Phi	0	0	0	0	s	0
1	s	0	0	0	0	0	6:Phi	0	0	0	0	0	0
0	0	0	0	0	0	0	7:p	0	0	0	0	0	1

We see that it has 13 columns, corresponding to 13 estimable parameters - we know from our initial analysis that this model does indeed have 13 estimable parameters. From this design matrix, we can build the other two models in the candidate model set ($\{\varphi_{g+t}p_{.}\}$, $\{\varphi_{g}p_{.}\}$) simply by deleting the appropriate columns from the design matrix (e.g., for the additive model $\{\varphi_{g+t}p_{.}\}$, we simply delete the interaction columns 8 → 12).

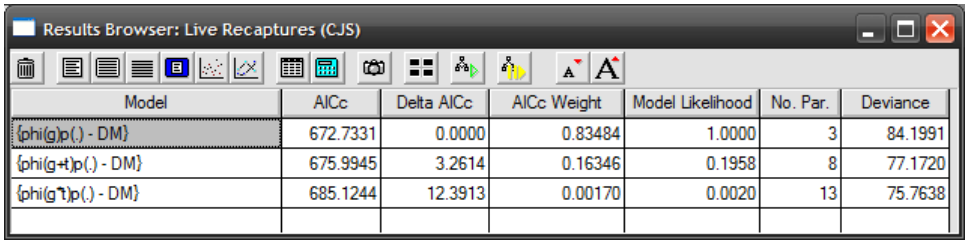
Here are the model fits for the 3 models, built using the individual covariates approach:



Results Browser: Live Recaptures (CJS)

Model	AICc	Delta AICc	AICc Weight	Model Likelihood	No. Par.	Deviance
{phi(g)p(.) - ind covar}	672.7331	0.0000	0.83484	1.0000	3	666.6762
{phi(g+t)p(.) - ind covar}	675.9945	3.2614	0.16346	0.1958	8	659.6491
{phi(g*t)p(.) - ind covar}	685.1244	12.3913	0.00170	0.0020	13	658.2409

Compare them with the results obtained using the standard approach where sex was treated as an 'attribute group':



Results Browser: Live Recaptures (CJS)

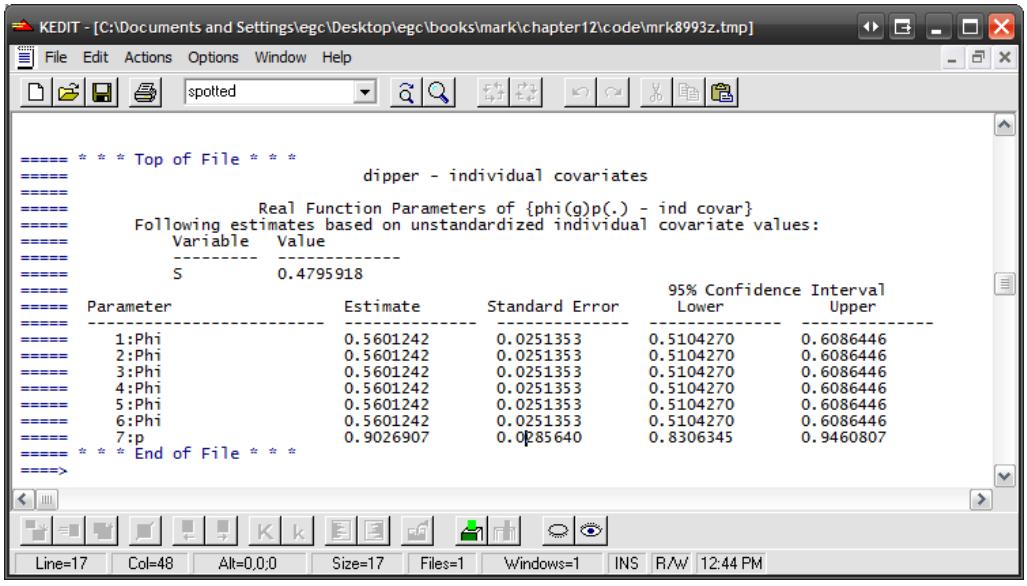
Model	AICc	Delta AICc	AICc Weight	Model Likelihood	No. Par.	Deviance
{phi(g)p(.) - DM}	672.7331	0.0000	0.83484	1.0000	3	84.1991
{phi(g+t)p(.) - DM}	675.9945	3.2614	0.16346	0.1958	8	77.1720
{phi(g*t)p(.) - DM}	685.1244	12.3913	0.00170	0.0020	13	75.7638

We see that the model AIC_c values, and the number of parameters, are identical between the two. However, the deviances are different. Does this indicate a problem? No - not if you think about it for a moment. If the AIC_c values and the number of parameters are the same, then the likelihoods for the models are also the same (since the AIC_c is simply a function of the sum of the likelihood and the

number of parameters - if two out of the three are the same between the different analyses, then so must the third (likelihood) be the same). In fact, if you look closely at the deviances, you'll see that the *difference* between the deviances - which is related to the likelihood (as discussed elsewhere) - is identical. For example, (666.6762 – 659.6491) = (84.1991 – 77.1720) = 7.0271.

So, the results are identical, regardless of the approach taken (attribute groups versus individual covariates coding for groups). And, it is pretty clear that the number of PIMs and the design matrix for the analysis using individual covariates is smaller (easier to handle, potentially less prone to errors) when using individual covariates. As such, is there any reason *not* to use the individual covariate approach to handling groups?

There are at least two possible reasons why you might not want to use the individual covariate approach for coding groups. First, as discussed earlier in this chapter, execution time generally increases for models involving individual covariates. For very large, complex data sets, this can be a significant issue. Second, and perhaps more important, while the individual covariate approach might simplify aspects of building the models, in fact it complicates derivation (reconstitution) of group-specific parameter estimates. For example, take estimates of φ from our simplest model, $\{\varphi_g p_.\}$. Using the standard attribute group approach, the estimates **MARK** reports for male and female survival are $\hat{\varphi}_m$ = 0.5702637 and $\hat{\varphi}_f$ = 0.5507352, respectively. What does **MARK** report for the estimates for this model fit using the individual covariates approach?

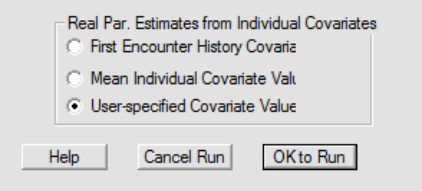


Clearly, the reported estimates using individual covariates appear to be quite different. But, are they? What does the value $\hat{\varphi}$ = 0.5601242 represent? What about the value 0.4795918 reported for the sex covariate, s? How can we reconstitute separate estimates of apparent survival for both males and females?

The key is remembering that this analysis is based on *individual* covariates. Recall that **MARK** defaults to reporting the parameter estimates for the mean value of the covariate. In this case, the sex covariate is 1 (indicating male) or 0 (indicating female). If the sex-ratio of the sample was exactly 50:50, then the mean value of the covariate would be 0.5. In fact, in the dipper data set, 47.96% of the individuals are male. Does that number look familiar? It should - it is the value of 0.4796 reported (above) as the average value of the covariate. And, the estimates of $\hat{\varphi}$ are the reconstituted values

of survival for an average individual. Thus, the value of 0.5601242 is essentially identical (to within rounding error) to the weighted average of $\hat{\varphi}_m = 0.5702637$ and $\hat{\varphi}_f = 0.5507352$, which we obtained from the analysis using attribute groups ($[0.4796 \times 0.5702] + [0.5204 \times 0.5507]) = 0.5601$) - here, the weights are the frequencies of males and females in the sample (i.e., the sex ratio of the sample).

OK - fine, but that still doesn't answer the practical question of how to reconstitute separate survival estimates for males and females? The 'brute-force' approach is to use a '**user-specified covariate value**', when you setup the numerical estimation. You do this by checking the appropriate radio button:



Now, when you click the '**OK to run**' button, **MARK** will ask you to specify the individual covariate value for that model - in this case, either a 1 (for male) or 0 (for female). If we enter a '1', run the model, and then look at the reconstituted parameter estimates, **MARK** shows $\hat{\varphi} = 0.5703$, which is exactly what we expected for males. Similarly, if instead we enter a '0' for the covariate value, **MARK** shows $\hat{\varphi} = 0.5507$ for females, again, precisely matching the estimate from the model fit using attribute groups.

OK, that is a functional solution, but not one that is particularly elegant (it can also be cumbersome if you have multiple levels of group, or a lot of interactions between one or more grouping variables and - say - time). It also is somewhat devoid of 'thinking', which is rarely a good strategy, since not understanding what **MARK** is doing when you 'click this button' or 'that button' will catch up with you sooner or later. The key to understanding what is going on is to remember from earlier in this chapter how parameter estimates were reconstituted for a given value of one or more individual covariates. Essentially, all you need to do is calculate the value of the parameter on the logit scale (assuming you're using the default logit link), and then back-transform to the real probability scale. For model $\{\varphi_g p_.\}$, the linear model is

$$\begin{aligned}\text{logit}(\hat{\varphi}) &= \beta_1 + \beta_2(s) \\ &= 0.2036416 + 0.0792854(s)\end{aligned}$$

So, if the value of the covariate is 1 (for males), then

$$\begin{aligned}\text{logit}(\hat{\varphi}_m) &= \beta_1 + \beta_2(s) \\ &= 0.2036416 + 0.0792854(1) \\ &= 0.282927\end{aligned}$$

which, when back-transformed from the logit scale to the normal probability scale,

$$\hat{\varphi}_m = \frac{e^{0.282927}}{1 + e^{0.282927}} = 0.570264$$

which is identical (within rounding error) to the estimate for male survival **MARK** reports using either the attribute group approach, or by specifying the value of the covariate in the numerical estimation using the individual covariate approach. The same is true for reconstituting the estimate for females.

While this is easy enough, it can get tiresome, especially if the linear model you're working with is

'big and ugly'. Even for fairly simple models like $\{\varphi_{g*t}p_{.}\}$, the linear model you need to work with can be cumbersome:

$$\begin{aligned}\text{logit}(\varphi) = \beta_1 + \beta_2(s) + \beta_3(t_1) + \beta_4(t_2) + \beta_5(t_3) + \beta_6(t_4) + \beta_7(t_5) \\ + \beta_8(s \cdot t_1) + \beta_9(s \cdot t_2) + \beta_{10}(s \cdot t_3) + \beta_{11}(s \cdot t_4) + \beta_{12}(s \cdot t_5)\end{aligned}$$

Each extra term in the equation adds to the possibility you'll make a calculation error. The complexity of the linear equation you need to work with will clearly be increased if you have > 2 levels of a grouping factor. We consider just such a situation in our final example.

11.7.2. Individual covariates for non-binary classification variables

Here, we consider the analysis of a simulated data set with 3 levels of some grouping variable (we'll call the grouping variable `colony`, and the three levels 'poor', 'fair', and 'good', reflecting the impact of some colony attribute on - say - apparent survival). The true model under which the simulated data (contained in `cjs3grp.inp`) were generated is model $\{\varphi_{g+t}p_{.}\}$ - additive survival differences among the 3 colonies (in fact, additive, and ordinal, such that $\varphi_g > \varphi_f > \varphi_p$, although this ordinal sequencing isn't of primary interest here). In the input file, the group columns (from left to right) indicate the `poor`, `fair` and `good` colonies, respectively. For our model set, we'll fit the same models (structurally) that we used for the dipper data used in the preceding example: $\{\varphi_{g*t}p_{.}\}$, $\{\varphi_{g+t}p_{.}\}$, $\{\varphi_{g}p_{.}\}$. Here are the results for the analysis of the data formatted using the attribute grouping approach:

Results Browser: Live Recaptures (CJS)

Model	AICc	Delta AICc	AICc Weight	Model Likelihood	No. Par.	Deviance
{phi(g+t)p(.) - DM}	21186.9257	0.0000	0.98007	1.0000	8	170.9553
{phi(g*t)p(.) - DM}	21194.7161	7.7904	0.01993	0.0203	16	162.7104
{phi(g)p(.) - DM}	21256.8935	69.9678	0.00000	0.0000	4	248.9325

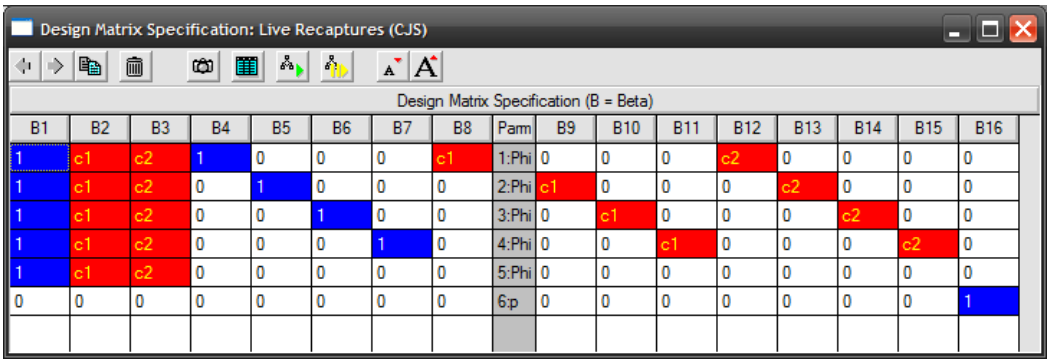
As expected, model $\{\varphi_{g+t}p_{.}\}$ has virtually all of the support in the data (it should, given that it was the true model under which the data were simulated in the first place).

Now, let's recast this analysis in terms of individual covariates. As noted in the preceding example, we need to specify enough covariates to correctly specify group association. Your first thought might be to use a single column, with (say) a covariate value of 1, 2 or 3 to indicate a particular colony. This would work, but the model you'd be fitting would be one where you'd be constraining the estimates to following a strict ordinal trend (this is strictly analogous to how you built trend models in Chapter 6). What if we simply want to test for heterogeneity among colonies? This, of course, is the null hypothesis of the standard analysis of variance. Since there are 3 colonies, then (perhaps not surprisingly) we need 2 columns of covariates to uniquely code for the different colonies. In effect, we're using *exactly* the same logic in constructing the covariate columns as we would in constructing corresponding columns in the design matrix. In fact, it is reasonable to describe what we're doing here - with individual covariates - as 'moving' the basic linear structure out the of the design matrix, and coding it explicitly in the input file itself.

We'll call the covariates `c1` and `c2`. For dummy coding of the colonies, we'll let '`1 0`' indicate the

first (poor) colony, '0 1' indicate the second (fair) colony, and '1 1' indicate the third (good) colony. So, the encounter history '111011 1 0 0' in the original file (indicating an individual from the poor colony) would be recoded as '111011 1 1 0'. Again, the first column after the encounter history after recoding is the frequency column, and is a '1' for all individuals (regardless of which colony they're in). The following two columns indicate values of the covariates c1 and c2, respectively. The reformatted encounter histories are contained in csj3ind.inp.

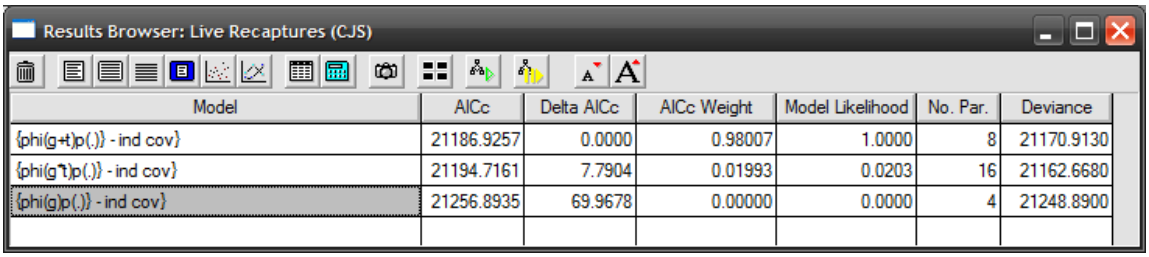
Now, when we specify the data type in **MARK**, we set the number of individual covariates to 2, and label them as c1 and c2, respectively. The design matrix corresponding to the most general model in the candidate model set $\{\varphi_{g*t}p_{.}\}$ is shown below:


Design Matrix Specification: Live Recaptures (CJS)

Design Matrix Specification (B = Beta)

B1	B2	B3	B4	B5	B6	B7	B8	Parm	B9	B10	B11	B12	B13	B14	B15	B16
1	c1	c2	1	0	0	0	c1	1:Phi	0	0	0	c2	0	0	0	0
1	c1	c2	0	1	0	0	0	2:Phi	c1	0	0	0	c2	0	0	0
1	c1	c2	0	0	1	0	0	3:Phi	0	c1	0	0	0	c2	0	0
1	c1	c2	0	0	0	1	0	4:Phi	0	0	c1	0	0	0	c2	0
1	c1	c2	0	0	0	0	0	5:Phi	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	6:p	0	0	0	0	0	0	0	1

Column 1 is the intercept, columns 2-3 are the covariates c1 and c2 (respectively), columns 4 → 7 are the time intervals (6 occasions, 5 intervals), columns 8 → 12 and 13 → 17 are the interactions of the covariates c1 and c2 with time, respectively. Column 18 is the constant encounter rate. Go ahead and fit this model to the data - notice immediately how much longer it takes **MARK** to do the numerical estimation (again, one of the penalties in using the individual covariate approach is the increased computation time required). Here are the results from this, and the other two reduced parameter models in the candidate model set:


Results Browser: Live Recaptures (CJS)

Model	AICc	Delta AICc	AICc Weight	Model Likelihood	No. Par.	Deviance
{phi(g+t)p(.)} - ind cov}	21186.9257	0.0000	0.98007	1.0000	8	21170.9130
{phi(g*t)p(.)} - ind cov}	21194.7161	7.7904	0.01993	0.0203	16	21162.6680
{phi(g)p(.)} - ind cov}	21256.8935	69.9678	0.00000	0.0000	4	21248.8900

If you compared these results with those shown on the preceding page (generated using group attributes rather than individual covariates), you'll see they are identical (again, the *differences* among the model deviances are identical, even if the individual model deviances are not). Again, using individual covariates in this case seems like a reasonable 'time-savings' strategy, since the number of PIMs, and the complexity of the general design matrix, is considerably reduced relative to what you'd face if you worked directly with attribute groups in the 'standard' way.

However, as noted in our discussion of the preceding dipper analysis, there are other potential 'costs' which might temper your enthusiasm for using the individual covariate approach to coding 'attribute

groups'. First, you'll need to handle reconstituting parameter estimates from what might potentially be pretty sizeable linear model (for our present example, it's sufficiently sizeable - 15 terms - that we won't write it out in full here). Second, *you* (instead of **MARK**) would have to handle the accompanying calculation of SE of the reconstituted estimates (using the Delta method - Appendix B). However, while this is possible (albeit somewhat time consuming), what is not possible is the derivation of the SE for the *effect size* (see Chapter 6 - section 6.12) for the difference between levels of a discrete 'attribute variable' when you've coded the 'attribute variable' using an individual covariate (e.g., '`sex`' - see section 11.7; the dipper example in subsection 11.7.1). Calculation of the SE for the 'effect size' (i.e., the difference between the estimates for different levels of the 'attribute variable') requires an estimate of the variance-covariance matrix between estimates for the different attribute levels, which is not estimable when using the individual covariate approach. Finally, generating model averaged parameter estimates from models with individual covariates is decidedly more complicated (as discussed in the next section) than for models without individual covariates.

So, while there is a clear 'up-front savings' in terms of simpler PIMs, and simpler design matrices, when using the individual covariate approach to handling attribute groups, the 'after-the-fact cost' of the number of things you'll need to do by hand (or, more typically, program into some spreadsheet) to generate parameter estimates is not insubstantial, and may be more than the hassle of dealing with lots of PIMs and big, ugly design matrices. An alternative to using individual covariates to simplify model-building is to use the **RMark** package (see Appendix C).

11.8. Model averaging and individual covariates

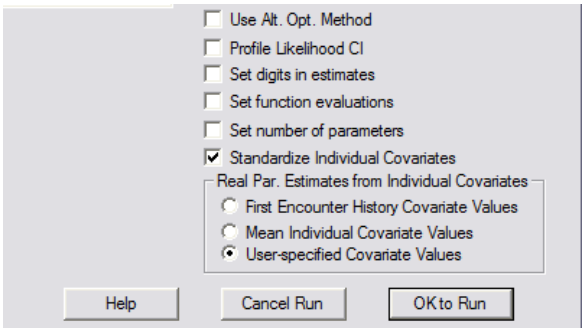
In chapter 4 we introduced the important topic of model averaging. If you don't remember the details, or the motivation, it might be a good idea to re-read the relevant sections. In a nutshell, the idea behind model averaging is pretty simple: there is uncertainty in our model set as to which model is 'closest to truth'. We quantify this uncertainty by means of normalized AIC weights-the greater the weight, the more support in the data for a given model in that particular model set. Thus, it seems reasonable that any average parameter value must take this uncertainty into account. We do this by (in effect) weighting the estimates over all models by the corresponding model weights (strictly analogous to a weighted average that you're used to from elementary statistics).

For models with individual covariates, you might guess that the situation is a bit more complex. The model averaging provides average parameter values over the models, but what you're often (perhaps generally) after with individual covariates is the 'average survival probability for an organism with a value of individual covariate XYZ'. For example, suppose you've done an analysis of the relationship of body mass to survival, using individual body mass as a covariate in your analysis. Some of your models may have body mass (`mass`) included, some may have `mass`, and `mass2` (as in the first example in this chapter). What would report as the 'average survival probability for an individual with body mass *X*'?

Mechanically, what you would need to do, if doing it by hand, is take the reconstituted values of φ for each model, for a given value of the covariate, then average them using the AIC weights as weighting factors (for models without the covariate, the β for the covariate is, in fact, 0). This is fairly easy to do, but a bit cumbersome by hand. Moreover, you have the problem of calculating the standard errors. Fortunately, **MARK** has an option to let you handle this 'drudgery' automatically. Basically, **MARK** lets you (the user) 'define' which individual covariate values are used in deriving the average parameter estimate.

Consider the following example - once again using `indcov2.inp`. Two models are included in the model set: $\{\varphi_{(t.mass)}p_{.}\}$ and $\{\varphi_{(t+mass)}p_{.}\}$ Again, `mass` is the individual covariate in this analysis. Now, we might want to know what the model-averaged survival probability is for a particular mass - say,

some value near the extremes of the range (a very light or very heavy individual), or perhaps the mean value. **MARK** makes it **very** easy to do this. Simply build the models, each time specifying whether you want **MARK** to provide real parameter estimates from either the first encounter record, a user-defined set of values, or the mean of the covariates. For purposes of demonstration, we'll use a user-defined covariate value (which allows us to generate a model-averaged estimate of survival for a covariate value we specify). Check the appropriate box (shown at the top of the next page), and then click the '**OK to run**' button for your model.



If you've checked the '**user-specified covariate values**' radio button, you'll then be presented with another small window asking you to enter the values of the covariates you want to generate real parameter estimates for. In this example, we have 2 covariates, `mass`, and `mass2`. So, the window will ask you to enter 2 different values. In the data used in this example, the weights are normalized to account for differences in age at time of measurement, so are expressed in terms of residuals from 0 (nothing to worry about, just something that made sense for these data). Using this scaling, a value of `mass`=120 reflects a fairly heavy (large) individual (the mean is 0 for normalized data, so any positive value of the covariate is > mean). Thus, `mass`=120, and `mass2` = 14,400. You enter both of these into the window when prompted by **MARK**.

That's it at this stage. We proceed to run this, and the other model in our model set. The resulting model fits are shown below:

Results Browser: Live Recaptures (CJS)

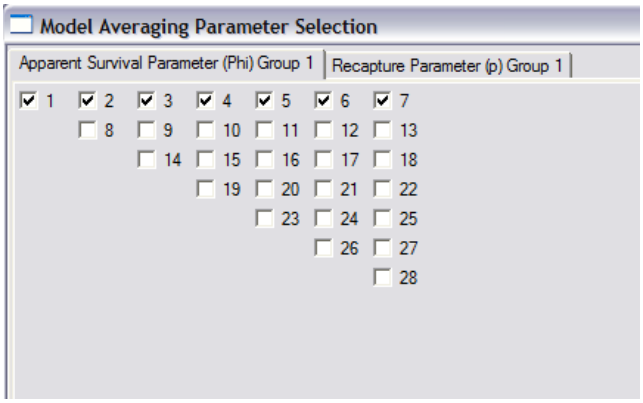
Model	AICc	Delta AICc	AICc Weight	Model Likelihood	No. Par.	Deviance
{phi(t+ mass)p(.)}	11814.1484	0.0000	0.77507	1.0000	9	11796.1200
{phi(t+mass+t.mass)p(.)}	11816.6227	2.4743	0.22493	0.2902	15	11786.5470

Before we can model average, we need to know a little about these models, and the analysis in general. As noted in the title bar of the browser, these data involve live encounter data only. The models are based on encounter data from individuals marked as adults. As such, we're interested in the survival over each interval in the study. To model average survival, given the values of the individual covariates of `mass`=120 and `mass2`=14,400, simply select the '**Output | Model Averaging**' option. This will present you with a large tabbed window (one tab for each parameter). We're interested in the survival parameter (φ) only. And, specifically, survival (φ), which corresponds to at least one element in each column of

the model averaging 'PIM'. So, we check off the first row (as shown at the top of the next page).

That's it, basically. **MARK** will then generate average survival values for the values of `mass`=120 and `mass2`=14,400, for each interval. That's it, really.

A cautionary note - while the preceding seems easy enough, there is at least one potential 'gotcha' that you need to be aware of. If you estimated the β parameters with standardized covariates, then you **must** enter standardized values for user-specified covariates. For example, if the β parameters are estimated with standardized covariates, then a user-specified value of 0.0 will correspond to the mean, and a user-specified value of 1.96 will correspond to a value 2 standard deviations above the mean.



11.9. GOF Testing and individual covariates

Well, now that we've seen how easy it is to handle individual covariates, now for the good news/bad news part of the chapter. The good news is that individual covariates offer significant potential for explaining some of the differences among individuals, which, as we know (see Chapter 5), is one potential source of lack of fit of the model to the data.

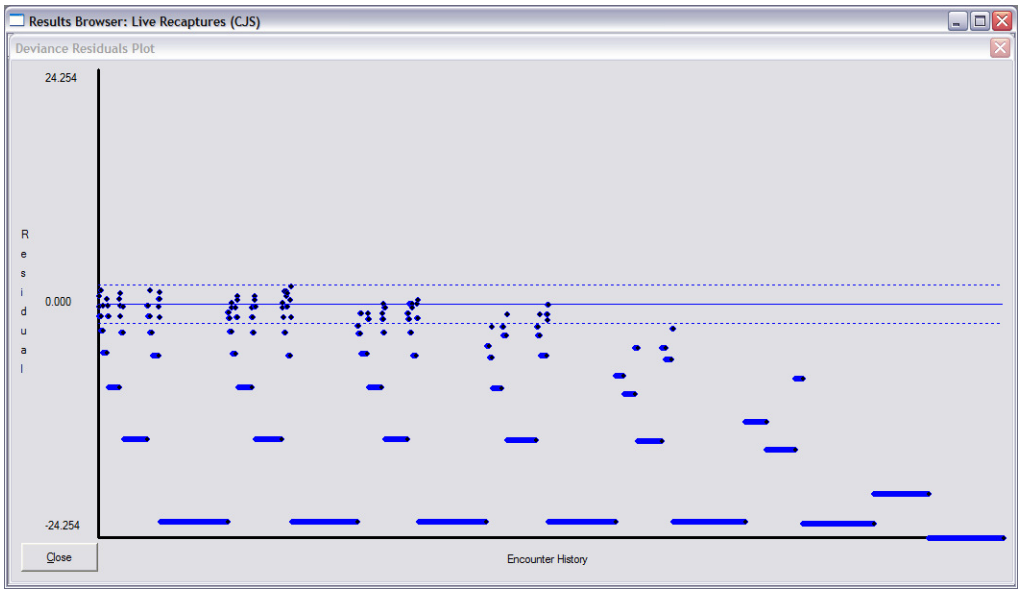
OK - now the bad news. At the moment, we don't have a good method for testing fit of models with individual covariates. For the moment, the recommended approach is to perform GOF testing on the most general model that does not include the individual covariates, and use the $\hat{c}$ value for this general model on all of the other models, even those including individual covariates. If individual covariates will serve to reduce (or at least explain) some of the variation, then this would imply that the $\hat{c}$ from the general model without the covariates is likely to be too high, and thus, the analysis using this $\hat{c}$ will be 'somewhat conservative'. So, keep this in mind...

begin sidebar

individual covariates and deviance plots

One approach to assessing the fit of a model to a particular set of data is to consider the *deviance residual plots*. While this can prove useful - in particular, to assess lack of fit because the structure of the model is not appropriate given the data (e.g., TSM models - see Chapter 7), if you try this approach for models with individual covariates, you'll quickly run into a problem.

For example, consider the deviance residual plot for the first example analysis presented in this chapter (for model $\{\varphi_{.}\, p_{.}\}$).



Clearly, something 'strange' is going on - we see fairly discrete 'clusters' of residuals, virtually all below the 0.000 line. Obviously, this is quite different than any other residual plot we've seen so far.

Why the difference? In simple terms, the reason that the residual plots change so much when an individual covariate is added is because the number of animals in each observation changes. Without individual covariates, the data are *summarized* for each unique capture history, so that variation within a history due to the individual covariate is lost. However, when the covariate is added into the model, each animal (i.e., each encounter history, even if it is the same as another history) is plotted as a separate point. The result is quite different, obviously. Without individual covariates, the binomial functions are the sample size, so animals are 'pooled'. With individual covariates, the number of animals is the sample size, each resulting in a unique residual.

In other words, the deviance residual plots for models with individual covariates are not generally interpretable.

end sidebar

11.10. Summary

That's it for Chapter 11! In this chapter, we looked at the basic mechanics of using **MARK** to fit models where one or more parameters are constrained to be functions of individual covariates. Individual covariates can be used with **any** of the models in **MARK** (not just recapture models). This is a significant increase in the flexibility of analyses you can execute with **MARK**.

CHAPTER 12

Jolly-Seber models in MARK

Carl James Schwarz, *Simon Fraser University*
A. Neil Arnason, *University of Manitoba*

The original Jolly-Seber (JS) model (Jolly, 1965; Seber, 1965) was primarily interested in estimating abundance. Since then, the focus of many mark-recapture experiments changed to estimating survival rates (but not abundance) using the Cormack-Jolly-Seber (CJS) models (Cormack, 1964; Jolly, 1965; Seber, 1965) particularly with the publication of Lebreton *et al*. (1992). In previous chapters concerning analysis of live encounter data, we have focussed exclusively on CJS models. In recent years, however, interest has returned to estimating parameters related to abundance such as population growth (λ_i), recruitment (f_i), as well as abundance (N_i).*

Much of the theory about estimating population growth, recruitment, and abundance can be found in Williams *et al*. (2002).

12.1. Protocol

The protocol for JS experiments is very similar to that of CJS experiments. In each of K sampling occasions, animals are captured. Unmarked animals are tagged with individually identifiable tags and released. Previous marked animals have their tag numbers read and are again released.† The key difference between JS and CJS experiments is the process by which unmarked animals are captured and marked. In CJS experiment, no assumptions are made about how newly marked animals are obtained. The subsequent process of recovering marked animals in CJS models is conditional upon the animal being released alive at first encounter, and survival and catchability refer only to these marked animals.‡ In JS experiments, the process by which unmarked animals are newly captured to be marked and released is crucial – the assumptions about this process allows the experimenter to estimate recruitment and population sizes. In particular, it is assumed that unmarked animals in the population have the same probability of capture as marked animals in the population, i.e., that newly captured unmarked animals are a random sample of all unmarked animals in the population.

* One of the reasons for preferring estimation of population growth is that estimates of population growth are fairly robust against heterogeneity in catchability (Schwarz, 2001), and tag loss (Rotella and Hines, 2005).

† Losses on capture are possible at every sampling occasion and are ignored in the discussion that follows.

‡ Of course, we hope that the survival of the marked subset of animals tells us something about the remaining unmarked animals in the population at large.

© Cooch & White (2014)

This assumption of equal catchability for marked and unmarked animals is needed to estimate abundance or recruitment or population growth and is required for the Pradel, Link-Barker, *POPAN*, and Burnham JS formulations in MARK...

Other assumptions about the experiment are similar to those for the CJS model:

- Animals retain their tags throughout the experiment.*
- Tags are read properly.
- Sampling is instantaneous.
- Survival probabilities are the same for all animals (marked and unmarked) between each pair of sampling occasions (homogeneous survival).
- Catchablity is the same for all animals (marked and unmarked) at each sampling occasion (homogeneous catchability). This is the most crucial assumption for JS models.†
- The study area is constant. If the study area changes over time, then the population size may change with the changing size of the study area.

There are generally two sources of non-closure in any particular study. Animals may leave the population through death or permanently emigrate. Conversely, animals may enter the study area from outside (immigration) or be recruited from within the study area (e.g. fish growing into the catchable portion of the population). Specific tests for closure have been developed (e.g., Stanley and Burnham, 1999), but more often tests for closure are performed by fitting models with no apparent mortality ($\varphi = 1$), or no apparent recruitment ($f = 0$, $\lambda = \varphi$, or $b = 0$), or both and letting the AIC_c indicate the appropriate weight for such simpler models.

12.2. Data

The basic unit of analysis is the capture history, a sequence of 0's and 1's that indicates when a particular animal was seen in the experiment. The JS models in **MARK** use the LLLLL capture history format. For example, the history (`011010`) indicates that an animal was captured for the first time at sampling occasion 2, was seen again at sampling occasion 3, not seen at sampling occasion 4, seen at sampling occasion 5, and not seen after sampling time 5.‡ Either individual or grouped capture histories may be used.

In many papers, the list of capture histories is too long to publish, and so a series of summary statistics are commonly used (Table 12.1; see also the reduced and full m-array descriptions in Chapter 5). For example, the history (`011010`) would contribute a count of 1 to n_2, n_3, n_5, u_2, m_3, m_5, R_2, R_3, R_5, r_2, r_3, and z_3.

* Refer to Cowen and Schwarz (2006) for dealing with tag loss in JS experiments.
† Refer to Pledger and Efford (1998) for details on dealing with heterogeneity in JS models.
‡ Again losses on capture are ignored for now but are handled in the same way as elsewhere in **MARK**.

Table 12.1: *Summary statistics often used for JS experiments. Losses-on-capture are found as* $n_i - R_i$.

Statistics	Definition
n_i	Number of animals captured at occasion i, both marked and unmarked. $n_i = m_i + u_i$.
u_i	Number of unmarked animals captured at occasion i.
m_i	Number of previously marked animals captured at occasion i.
R_i	Number of animals released alive at occasion i^+, i.e., just after sampling occasion i.
r_i	Number of animals from R_i that are subsequently captured after occasion i.
z_i	Number of animals seen before i, seen after occasion i, but not seen at occasion i.

While these summary statistics form the sufficient statistics for the Jolly-Seber probability model, their use has fallen out of favor in place of the raw histories used by **MARK** for two reasons. First, the use of individual covariates will require the individual capture history vectors (see Chapter 2, and Chapter 11). Second, it is difficult to compute goodness-of-fit statistics (i.e., the **RELEASE** suite of tests; Chapter 5) from the summary statistics.* If only summary statistics are available, it is possible to work 'backwards' and create a set of histories that will reproduce these summary statistics that can be used with **MARK** to fit various models. One problem in using these pseudo-histories is that goodness-of-fit tests are nonsensical – the goodness-of-fit tests require the full capture history of each animal.

12.3. Multiple formulations of the same process

There are a number of formulations used in **MARK** to estimate abundance and related parameters, e.g., the *POPAN*; the Link-Barker and Pradel-recruitment; and the Burnham JS and Pradel-λ formulations. All of these models are slightly different parameterizations of the underlying population processes, and all are (asymptotically) equivalent in that they should give the same estimates of abundance and related parameters. The two main differences among the various formulations are

1. the way in which they parameterize new entrants to the population
2. if estimation is conditional upon the animals actually seen in the study (refer to Sanathanan 1972, 1977).

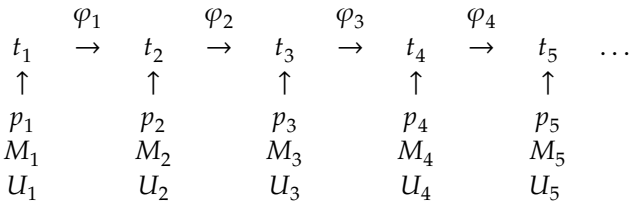
All of the formulations model the recapture of marked animals in the same way. In this section, several of these models will be examined and contrasted.

* Indeed, if the summary statistics are used by themselves, the fully time dependent JS models will be 'perfect' fit to the summary statistics regardless if the model overall is a good fit.

12.3.1. The Original Jolly-Seber formulation

In the original JS formulation of Jolly (1965) and Seber (1965), the population process can be modeled as shown in Figure 12.1. The parameters p_i and φ_i are similar, but not identical to those in the CJS models. The parameter p_i is the probability of capture of both unmarked and marked animals that are alive at occasion i (the CJS models referred only to marked animals); the parameter φ_i refers to the survival probabilities of both marked and unmarked animals between occasions i and $i+1$ (the CJS models referred only to marked animals).

Figure 12.1: *Original process model for JS experiments. p_i represents the probability of capture at occasion i; φ_i represents the probability of an animal surviving between occasions i and $i+1$; and M_i and U_i represent the number of marked and unmarked animals alive at occasion i. Losses-on-capture are not modeled here, but are easily included.*



The number of marked animals in the population just before occasion $i+1$ is found as $M_{i+1} = (M_i + u_i)\varphi_i$ where u_i is the number of newly unmarked animals captured and subsequently marked.

The number of *net* new entrants to the population was *defined* as

$$B_i = U_{i+1} - \varphi_i(U_i - u_i)$$

The B_i values refer to the *net* number of new entrants to the population between sampling occasion i and occasion $i+1$. The reference to '*net*' number of new entrants implies that animals that enter between two sampling occasions but then die before being subject to capture at occasion $i+1$ are excluded.* As in the CJS models, the term *survival* refers to *apparent* survival – permanent emigration is indistinguishable and treated the same as mortality. Similarly, the term *births* refers to any new animals that enter the study population regardless if *in situ* natural births or immigration from outside the study area.

The likelihood function consists of three parts. The first part models *losses-on-capture* using a simple binomial distribution as in the CJS models. The second part models the *recapture of marked animals* in exactly the same way as in the CJS model. Finally, the third part models *the number of unmarked animals captured at occasion i* as a binomial function of the number of unmarked animals in the population, i.e., u_i is $Binomial(U_i, p_i)$.

The estimates of p_i and φ_i are found in exactly the same way as in the CJS models. The estimated unmarked population sizes were estimated as $\hat{U}_i = u_i/\hat{p}_i$. The estimated number of births was found by substituting in the estimates in the previous definition and does not form part of the likelihood. Finally, the estimates of population size at each time point is found by adding the estimates of U_i and M_i.

* The term *gross* number of entrants would include these deaths prior to the next sampling occasion. Refer to Schwarz *et al.* (1993) for details on the estimation of these *gross* births.

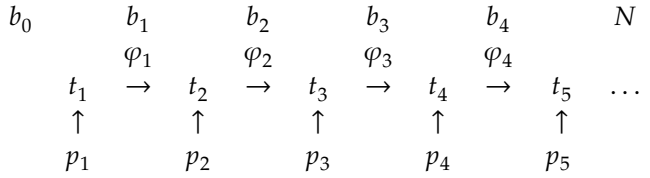
12.3.2. *POPAN* formulation

Schwarz and Arnason (1996) adopted a slightly different parameterization, for a number of reasons:

- The parameters B_i never directly entered into the likelihood function. The number of entrants must be non-negative, but it was difficult to enforce $\hat{B}_i \geq 0$ and negative estimates of births were often obtained.
- Because the B_i did not appear in the likelihood, how could these be forced to be equal across groups following the Lebreton *et al*. (1992) framework?
- How could death-only models (e.g., all B_i known to be zero) or birth-only models (all φ_i=1) or closed models be obtained by constraining the likelihood function.

In their parameterization, first implemented in the computer package *POPAN* and now a sub-module of **MARK**, they postulated the existence of a *super-population* consisting of all animals that would ever be born to the population, and parameters b_i which represented the probability that an animal from this hypothetical super-population would enter the population between occasion i and $i+1$ as shown in Figure 12.2.*

Figure 12.2: *Process model for POPAN parameterization of JS experiments. p_i represents the probability of capture at occasion i; φ_i represents the probability of an animal surviving between occasions i and $i+1$; and b_i represents the probability that an animal from the super-population (N) would enter the population between occasions i and i+1 and survive to the next sampling occasion i+1. Losses-on-capture are assumed not to happened, but are easily included.*



Now the expected number of *net* new entrants is simply found as $E[B_i] = Nb_i$. If B_0 represents the number of animals alive just prior to the first sampling occasion, then

$$N = B_0 + B_1 + B_2 + \cdots + B_{K-1}$$

i.e., the total number of animals that ever are present in the study population. The parameters b_i are referred to as *PENT* (Probability of Entrance) probabilities in **MARK**. Notice that $b_0 + b_1 + \cdots + b_{K-1} = 1$; – this will have consequences later when the models are fitted using **MARK**. Even though the number of new animals is not modeled in the process, modeling the entrance probabilities and a super-population size is equivalent.

* The super-population approach was first described by Crosbie and Manly (1985) where distribution functions (e.g. a Weibull distribution) was used to model survival time once an animal had entered the population. To our knowledge, there is no readily available computer code for the Crosbie and Manly (1985) model.

Under this parametrization,

$$\begin{aligned} E[N_1] &= Nb_0 \\ E[N_2] &= E[N_1]\varphi_1 + Nb_1 \\ &\vdots \end{aligned}$$

The probability of any capture history can be expressed using these parameters. For example, Pr[(01010)] is found as:

$$\Pr[(01010)] = [b_0(1-p_1)\varphi_1 + b_1]\, p_2\varphi_2(1-p_3)\,\varphi_3 p_4\,[1-\varphi_4+\varphi_4(1-p_5)]$$

As in the CJS models, the fate of the animal after the last capture is unknown – either it died, or it survived and was not seen at occasion 5. In a symmetrical fashion, the fate of the animal before the first time it is captured is also unknown. Either it was present in the population prior to sampling occasion 1 and wasn't seen at occasion 1 and survived to occasion 2, or it entered the study population between sampling occasions 1 and 2 and survived to sampling occasion 2 where it was captured for the first time. The likelihood function is again a multinomial function over all the observed capture histories. Schwarz and Arnason (1996) showed that it could be factored into three parts:

$$\mathcal{L} = \Pr(\text{first capture}) \times \Pr(\text{subsequent recaptures}) \times \Pr(\text{loss on capture})$$

where the second and third components are identical to the CJS models. It turns out that similar to CJS models, not all parameters are identifiable and only functions of parameters can be estimated in the fully time-dependent model. The set of non-identifiable parameters is given in Table 12.2. In particular, the final survival and catchability parameters are confounded (as in the CJS models), and symmetrically the initial entrance and catchability are confounded. This impacts three other sets of parameters, in particular N_1 and N_K cannot be cleanly estimated, nor can b_1 and b_{K-1}. If confounding takes place, the estimated super-population number may be suspect, so some care must be taken in fitting appropriate models. For example, models with equal catchability over sampling occasions make all parameters identifiable.

This confounding implies that careful parameter counting may have to be done when fitting *POPAN* models. The fully time-dependent model $\{p_t, \varphi_t, b_t\}$ has K parameters for catchability, $K-1$ parameters for survival, K parameters for the *PENT*s, and 1 parameter for the super-population size for a total of $3K$ parameters. However, not all are identifiable and the *PENT*s must sum to one. Only the products $b_0 p_1$ and $\varphi_{K-1}p_K$ can be estimated, and one of the *PENT*s is not 'free' (as the sum must equal 1), leaving $3K-3$ parameters that can be estimated for each group.

Furthermore, as indicated in Table 12.2, the b_1 and b_{K-1} parameters are affected (the estimates reflect the combination of parameters as listed in the table) which further affect N_1 and N_K. While the latter parameter combinations are 'estimable', they seldom represent anything biologically useful. The actual number of parameters reported by **MARK** in the results browser should be checked carefully.

Table 12.2: *Confounded parameters in the POPAN parameterization in the fully time-dependent model. In order to resolve this confounding, the models must make assumptions about the initial (p_1) and final (p_K) catchabilities. For example, a model may assume that catchabilities are equal across all sampling occasions.*

Function	*Interpretation*
$\varphi_{K-1}p_K$	Final survival and catchability.
$b_0 p_1$	Initial entrance and catchability.
$b_1 + b_0(1-p_1)\varphi_1$	Entry between first and second occasions cannot be cleanly estimated because initial entrance probability cannot be estimated. **MARK** (and other programs) will report an estimate for this complicated function of parameters but it may not be biologically meaningful.
b_{K-1}/φ_{K-1}	Entry prior to last sampling occasion cannot be cleanly estimated because final survival probability cannot be estimated. **MARK** (and other programs) will report an estimate for this complicated function of parameters but it may not be biologically meaningful.

Once estimates of p, φ, b, and N are obtained, the estimated number of births is obtained as $\hat{B}_i = \hat{N}\hat{b}_i$. The estimated population sizes are obtained in an iterative fashion:

$$\begin{aligned}\hat{N}_1 &= \hat{B}_0 \\ \hat{N}_2 &= \hat{N}_1\varphi_1 + \hat{B}_1 \\ &\vdots\end{aligned}$$

If losses on capture occur, they are removed before the population size at occasion i is propagated to occasion $i+1$.

The likelihood does not contain any terms for B_i or N_i – these are derived parameters and standard errors for these estimates are found using the Delta method (see Appendix 2).

12.3.3. Link-Barker and Pradel-recruitment formulations

The Link-Barker (2005) and Pradel-recruitment* (1996) formulations are conceptually the same and the process model is shown in Figure 12.3. The parameters for survival (φ_i) and catchability (p_i) are standard. The parameter f_i is interpreted as a *per capita* recruitment probability , i.e., how many net new animals per animal alive at occasion i enter the population between occasion i and $i+1$?

Unlike the *POPAN* formulation, the Link-Barker formulation conditions upon an animal being seen somewhere in the experiment. This eliminates the necessity of estimating the super-population size, but also means that abundance cannot be directly estimated. Any probability of a history must be normalized by the probability of being a non-zero history.

* There are three different Pradel models and '-recruitment' refers to the Pradel models parameterized using f_i terms

Figure 12.3: *Process model for Link-Barker and Pradel-recruitment parameterization of JS experiments. p_i represents the probability of capture at occasion i; φ_i represents the probability of an animal surviving between occasions i and $i+1$; and f_i represents the net recruitment probability , i.e., the per capita number of new animals that enter between occasions i and $i+1$ and survive to the next sampling occasion $i+1$ per animal alive at occasion i. Losses-on-capture are assumed not to have happened, but are easily included.*

$$\begin{array}{ccccccccccc} & & f_1 & & f_2 & & f_3 & & f_4 & & \\ & & \varphi_1 & & \varphi_2 & & \varphi_3 & & \varphi_4 & & \\ t_1 & \rightarrow & t_2 & \rightarrow & t_3 & \rightarrow & t_4 & \rightarrow & t_5 & \ldots \\ \uparrow & & \uparrow & & \uparrow & & \uparrow & & \uparrow & & \\ p_1 & & p_2 & & p_3 & & p_4 & & p_5 & & \end{array}$$

For example, Pr[(01010)|animal seen] is proportional to:

$$\Pr\left[(01010)|\text{animal seen}\right] \propto \left[(1-p_1)\varphi_1 + f_1\right]/p_1 \times p_2\varphi_2\left(1-p_3\right)\varphi_3 p_4\left[1-\varphi_4+\varphi_4\left(1-p_5\right)\right]$$

where the constant of proportionality is related to the probability of seeing an animal somewhere in the experiment. As in the CJS models, the fate of the animal after the last capture is unknown – either it died, or it survived and was not seen at occasion 5. In a symmetrical fashion, the fate of the animal before the first time it is captured is also unknown – either it was present among animals seen in the experiment at time 1, was not seen, and survived to time 2, or it entered between times 1 and 2. The likelihood function is a multinomial function over all the observed capture histories conditional upon an animal being seen somewhere in the experiment.

The implementation of the Link-Barker model differs from the Pradel-recruitment formulation in a number of ways. First, Link-Barker partitioned the likelihood in a similar fashion to the *POPAN* formulation which made it easier to implement in the Bayesian context of their paper. Second, Link-Barker explicitly modeled the confounded parameters (but the **MARK** implementation leaves the confounded parameters separate and it is the user's responsibility to understand the confounding). Third, losses on capture are handled differently between the two formulations and this affects the interpretation of the recruitment parameters.

The Link-Barker model also differs from the *POPAN* formulation as there is no need to postulate the existence of a super-population – the model is fit conditional upon the observed number of animals in the experiment.* Section 12.3.5 outlines the equivalences between the Link-Barker parameters and those of other formulations.

As in the *POPAN* formulation, the fully time-dependent Link-Barker and Pradel-recruitment models have a number of parameter confoundings as listed in Table 12.3. On the surface, the fully time-dependent model $\{p_t, \varphi_t, f_t\}$ has K catchability parameters, $K-1$ survival parameters, and $K-1$ recruitment parameters for a total of $3K-2$ parameters. However, only the product $\varphi_{K-1}p_K$ and the ratio f_1/p_1 can be estimated, leaving a net of $3K-4$ parameters. The estimate for f_{K-1} estimates a function of other parameters as shown in Table 12.3.

* The implementation of Schwarz and Arnason (1996) in the *POPAN* package also estimates parameters conditional upon being seen, and then adds another step to estimate the super-population size.

Table 12.3: *Confounded parameters in the Link-Barker parameterization in the fully time-dependent model. In order to resolve this confounding, the models must make assumptions about the initial (p_1) and final (p_K) catchabilities. For example, a model may assume that catchabilities are equal across all sampling occasions.*

Function	*Interpretation*
$\varphi_{K-1}p_K$	Final survival and catchability
$(\varphi_1 + f_1)/p_1$	Initial recruitment and survival
$f_{K-1}p_K$	Final recruitment and catchability cannot be cleanly estimated. **MARK** (and other programs) will report an estimate for this complicated function of parameters but it may not be biologically meaningful.

The abundance at each sampling occasion and the absolute number of new entrants cannot be estimated even as derived parameters because of the conditioning upon animals seen at least once during the experiment.

12.3.4. Burnham JS and Pradel-λ formulations

The final formulations to be considered in this chapter model new entrants to the population indirectly by modeling the rate of population growth (λ) between each interval where population growth is the net effect of survival and recruitment. If φ_i is the decrease in the population per member alive at time i, and f_i is the increase in the population per member alive at time i, then the sum of their contributions is the net population growth:

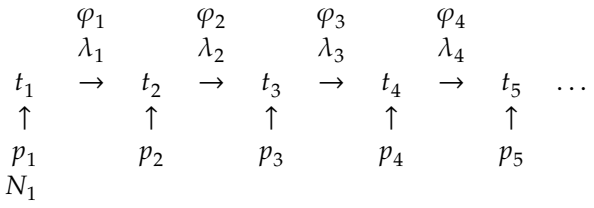
$$\lambda_i = N_{i+1}/N_i = \varphi_i + f_i$$

These formulations were developed by Burnham (1991) and Pradel (1996).

The key difference between the two parameterizations is that the Pradel-λ approach is conditional upon animals being seen during the study, while the Burnham JS formulation is not. Therefore, the Burnham Jolly-Seber formulation also includes a parameter for the population size at the start of the experiment. This enables the estimation of the population size at each subsequent time point. However, in practice, it is often difficult to get the Burnham-JS model to convergence during the numerical maximization of the likelihood. Although the implementation of this model has been thoroughly checked and found to be correct, **MARK** has some difficulty obtaining numerical solutions for the parameters because of the penalty constraints required to keep the parameters consistent with each other.* For this reason, only the Pradel-λ formulation will be discussed further in this section (and treated in depth in Chapter 13). The process model is shown in Figure 12.4. The parameters for survival (φ_i) and catchability (p_i) are standard. The parameter λ_i is interpreted as the ratio of successive population abundances.

* This convergence problem disappears for some models if simulated annealing is used for the numerical optimization. - J. Laake & E. G. Cooch, *pers. obs.*

Figure 12.4: *Process model for Burnham and Pradel-λ parameterization of JS experiments. p_i represents the probability of capture at occasion i; φ_i represents the probability of an animal surviving between occasions i and $i+1$; and λ_i represents the rate of population change. The population size at time 1, N_1 is used by the Burnham formulation, but not by the Pradel-λ formulation. Losses-on-capture are assumed not to happened, but are easily included.*



Unlike the *POPAN* and Burnham formulations, the Pradel-λ formulation conditions upon an animal being seen somewhere in the experiment. This eliminates the necessity of estimating the population sizes at any sampling occasion. But, the probability of a history must now be normalized by the probability of being a non-zero history. For example, Pr[(01010)|animal seen] is proportional to:

$$\Pr\left[(01010)|\text{animal seen}\right] \propto \left[\lambda_1 - p_1\varphi_1\right] \times p_2\varphi_2\left(1-p_3\right)\varphi_3 p_4\left[1-\varphi_4+\varphi_4\left(1-p_5\right)\right]$$

where the constant of proportionality is related to the probability of seeing an animal somewhere in the experiment. As in the CJS models, the fate of the animal after the last capture is unknown – either it died, or it survived and was not seen at occasion 5. In a symmetrical fashion, the fate of the animal before the first time it is captured is also unknown – either it was present at the initial sampling occasion and not seen, or was part of the population growth (over and above survival from the first sampling occasion). The likelihood function is a multinomial function over all the observed capture histories conditional upon an animal being seen somewhere in the experiment.

Section 12.3.5 outlines the equivalences between the Pradel-λ parameters and those of other formulations.

As in the *POPAN* formulation, the fully time-dependent Pradel-λ formulation has a number of parameter confoundings as listed in Table 12.4. On the surface, the fully time-dependent model $\{p_t, \varphi_t, \lambda_t\}$ has K catchability parameters, $K-1$ survival parameters, and $K-1$ growth parameters for a total of $3K-2$ parameters. However, only the product $\varphi_{K-1}p_K$ and the function $\lambda_1 - \varphi_1 p_1$ can be estimated, leaving a net of $3K-4$ parameters. Furthermore, the estimate for λ_{K-1} estimates a function of other parameters and may not be interpretable.

The abundance at each sampling occasion and the absolute number of new entrants cannot be estimated even as derived parameters because of the conditioning upon animals seen at least once during the experiment.

Table 12.4: *Confounded parameters in the Pradel-λ parameterization in the fully time-dependent model. In order to resolve this confounding, the models must make assumptions about the initial (p_1) and final (p_K) catchabilities. For example, a model may assume that catchabilities are equal across all sampling occasions.*

Function	*Interpretation*
$\varphi_{K-1}p_K$	Final survival and catchability
$\lambda_1 - \varphi_1 p_1$	Initial growth and survival
$\lambda_{K-1}p_K$	Final recruitment and catchability cannot be cleanly estimated. **MARK** (and other programs) will report an estimate for this complicated function of parameters but it may not be biologically meaningful.

Because population growth (λ) is a function both of survival and recruitment (i.e., $\lambda_i = \varphi_i + f_i$), the modeler should be careful about fitting simpler models that restrict population growth but leave survival time-dependent. For example the model $\{ p_t, \varphi_t, \lambda_\bullet \}$ would imply that recruitment varies in a time dependent fashion to exactly balance changes in survival to keep population growth constant. This may not be a sensible biological model.

Another potential problem with the Pradel-λ model is that no constraints are imposed in **MARK** that population growth must exceed the estimated survival probability. Consequently (as seen in the examples that follow), it is possible to get estimated survival probabilities of 80% while the estimated population growth rate is only 70%. This logically cannot happen, and is often an indication that recruitment did not occur in that interval – the illogical estimates are artifacts of the estimation process. Under these circumstances, models that separate recruitment from population growth may be preferred.

12.3.5. Choosing among the formulations

All of the formulations use the same input file in the same format. Which JS formulation should be used for a particular experiment? There are two considerations.

First, only certain of the formulations can be used in **MARK** if losses-on-capture occur in the experiment.

Secondly, and more importantly, different formulations give you different types of information and can be used to test different hypotheses. All of the formulations should give the same estimates of survival and catchability as all formulations estimate these from recaptures of previously marked animals using a CJS likelihood component. Even though all the models give different types of estimates for growth or recruitment or births, it is always possible to transform the estimates from one type to another by simple transformation and the standard errors can be found using the delta method.

The major equivalents are between NET births, recruitment, and population growth parameters.

Recruitment parameters are the net number of new animals that enter the population between occasions i and $i+1$ per animal present in the population at occasion i

$$f_i = \frac{B_i}{N_i}$$
$$= N\frac{b_i}{N_i}$$

Population growth is the proportionate increase in abundance between occasions i and $i + 1$:

$$\lambda_i = \frac{N_{i+1}}{N_i} = \frac{N_i\varphi_i + B_i}{N_i} = \varphi_i + f_i$$

Actual estimates from fitted models may not follow these exact relationships for several reasons.

First, certain estimates (e.g., apparent survival) should be constrained to lie between 0 and 1. If an estimated survival hits against this boundary, estimates of survival prior to and after this sampling occasion will also be affected. If the *identity* link of **MARK** is used, estimates are allowed to fall outside these 'normal' boundaries, and usually the exact relationships above then hold true among the estimates as well.

Second, losses on capture complicate the equivalences among parameters. For example, Link and Barker (2005) indicate that their f_i should be interpreted as the number of new animals that enter between occasions i and $i + 1$ per hypothetical animals alive at occasion i in the absence of losses on capture, while the Pradel-recruitment f_i is the number of new animals after losses on capture have been taken into account.

Lastly, **MARK** does **not** impose constraints that estimated population growth parameters must be at least as great as estimated survival probabilities. Consequently, it is possible (as seen in the examples) that the estimated population growth rate is less than the estimated survival probability which would imply a negative recruitment. In my opinion, formulations that model recruitment and survival as separate processes are preferred in these case – the estimated population growth rate can always be derived from these alternative models.

It cannot be emphasized too strongly, that **all of the formulations require the same careful attention to study design** – in particular the study area must remain a consistent size, and the probability of capturing an unmarked individual must be the same as a marked individual at each sampling occasion. A Cormack-Jolly-Seber experiment where marked animals are captured and released haphazardly, should not be then analyzed using any of the formulations of the Jolly-Seber model discussed in this chapter.

Table 12.5: *Summary of criteria to choose among the different JS formulations*

	losses on	estimates available for			
formulation	capture	*abundance*	*net births*	*recruitment*	λ
POPAN	yes	yes	yes	no	no
Link-Barker	yes	no	no	yes	yes
Pradel-recruitment	no	no	no	yes	yes
Burnham JS	yes	yes	yes	no	yes
Pradel-λ	yes	no	no	no	yes

• The implementation of Burnham's JS model in **MARK** often does not converge, and is not recommended (although convergence problems may be minimized for some models if simulated annealing is used for the numerical optimization. - J. Laake & E.G. Cooch, *pers. obs.*)

• The standalone package of *POPAN* will estimate recruitment, and population growth as derived parameters

12.3.6. Interesting tidbits

There have been a number of queries in the **MARK** forum (`http://www.phidot.org/forum`) about the use of *POPAN* and other models to estimate abundance. This section will try to answer some of these queries in more detail.

Deviance of 0 in *POPAN*

The following query was received in the **MARK** forum (`http://www.phidot.org/forum`, 2006-04-07) which asked:

> *'I read on one of the other posts that getting a deviance of zero was possible using the robust design model because the saturated model hadn't been computed yet and some constants were left out for faster computation. Is something along these lines at play in POPAN also because when I run a particular set of data, I get a deviance of zero?'*

The deviance of models is often computed as the negative of twice the difference in the log-likelihood between the current model and a 'saturated' model. The usual saturated model in CJS and other models that condition upon an animal's first capture, is to have a separate probability for each observed history, i.e., if ω is a capture history (e.g., `010011`) then the $\Pr(\omega)$ under the saturated model is found as $\Pr(\omega) = \frac{n_\omega}{n_{obs}}$ where n_ω is the number of animals with capture history ω, and n_{obs} is the total number of animals observed. In such cases, the log-likelihood of the saturated model (ignoring constants) is then

$$\sum n_\omega \log(p_\omega) = \sum n_\omega \log\left(\frac{n_\omega}{n_{obs}}\right)$$

However, this doesn't work for models where abundance is estimated because you need to include the animals with history (`000000`....), i.e., those animals not observed. This can only be computed if the population size is estimated which cannot be estimated under the saturated model. Hence a 'deviance' cannot be directly computed.

However, the likelihood can be portioned as shown earlier into components representing the probability of first capture, the probability of subsequent recapture given the animal has been captured, and the probability of losses-on-capture given that an animal is captured. Schwarz and Arnason (1996) and Link and Barker (2005) showed that the first component is essentially non-informative about the capture, survival, and loss-on-capture rates.

This suggests that an approximate deviance could be computed using only the latter two components, i.e., by conditioning upon animals that are seen at least once in the experiment. The difference between the two likelihoods would be based only on the part representing animals seen at least once. In practice, we would suggest that you compute a deviance based on conditioning on the observed animals. The easiest way is to use the Link-Barker model (which doesn't estimate abundance) but is 'equivalent' to the POPAN model. So if you fit a $\{p_t, \varphi_t, b_t\}$ model in *POPAN* look at the deviance of the $\{p_t, \varphi_t, f_t\}$ model in Link-Barker formulation. This could be used to estimate a variance-inflation factor to adjust reported standard errors.

12.4. Example 1 - estimating the number of spawning salmon

After spending several years at sea, coho salmon (*Oncorhynchus kisutch*) return to spawn in the Chase River, British Columbia. The normal life cycle of coho salmon is to return at age 3 as adults to spawn

and die. But, some precocious males return earlier at age 2 to spawn and die. One question of interest is if the distribution of salmon that return to spawn at different parts in the spawning period the same for regular adult and precocious males?

As fish return to the Chase River, they are captured using electrofishing gear. If they are unmarked they are given a unique tag number and released. If they were previously marked, the tag number is read. The experiment took place over a 10 week period in 1989, but the data from weeks 1 and 2, and weeks 9 and 10 were pooled and labeled as weeks 1.5 and 9.5. Approximately the same amount of effort was expended in each week of sampling. More details of this experiment are found in Schwarz *et al*. (1993).

The datafile is given in the `chase.inp` file in the sample data directory and a portion of it is reproduced in Figure 12.5. This study has two groups, the regular adult males and the precocious males (called jacks). Notice that a separate capture history is used for *each* tagged fish - unfortunately, this implies that the residual plots and deviance plots in **MARK** cannot be used to assess goodness of fit.

Figure 12.5: *Portion of the data for the Chase 1989 experiment*

```
/* Estimating salmon numbers returning to spawn in Chase River 1989 */
/* These are the male salmon with two groups.  */
/* Group1 = adults . group2=jacks */
/* Survey conducted over 10 weeks. Weeks 1 & 2 pooled. weeks 9 & 10 pooled */
11000000 -1 0  ;  /* tagnum=1  */
10000000 0 1  ;  /* tagnum=3  */
10000000 0 1  ;  /* tagnum=4  */
10000000 0 1  ;  /* tagnum=6  */
11000000 0 1  ;  /* tagnum=8  */
10110000 0 1  ;  /* tagnum=9  */
10000000 0 1  ;  /* tagnum=10  */
10000000 1 0  ;  /* tagnum=11  */
  ... additional histories follow ....
```

Summary statistics for this experiment are presented in Table 12.6.

Table 12.6: *Summary statistics for the Chase 1989 experiment*

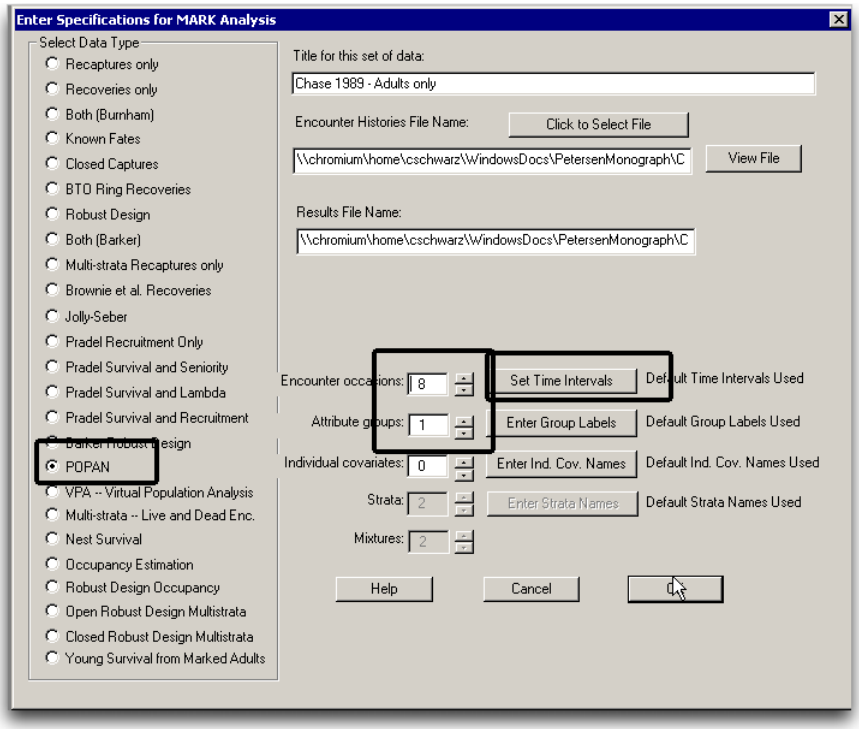
statistics for adults							statistics for jacks						
t_i	n_i	m_i	u_i	R_i	r_i	z_i	t_i	n_i	m_i	u_i	R_i	r_i	z_i
1.5	37	0	37	37	12	0	1.5	67	0	67	62	21	0
3.0	22	6	16	21	13	6	3.0	28	9	19	25	7	12
4.0	52	7	45	41	19	12	4.0	46	6	40	44	9	13
5.0	56	17	39	54	26	14	5.0	47	12	35	45	5	10
6.0	46	26	20	38	8	14	6.0	25	9	16	24	3	6
7.0	28	16	12	20	2	6	7.0	16	6	10	12	1	3
8.0	22	3	19	16	2	5	8.0	7	1	6	5	1	3
9.5	10	7	3	0	0	0	9.5	7	4	3	0	0	0

Because $n_i > R_i$ for some sampling occasions, this indicates that some losses-on-capture occurred. For example, there was one adult loss-on-capture in week 3; 11 adults lost in week 4, etc.

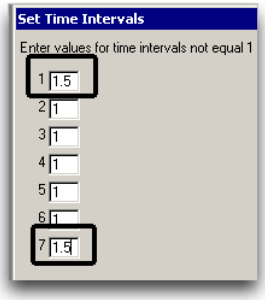
12.4.1. *POPAN* formulation

The *POPAN* super-population is a natural way to think of this experiment - a pool of fish is returning to spawn. During each week, a certain fraction of these returning fish decide to enter the spawning areas.

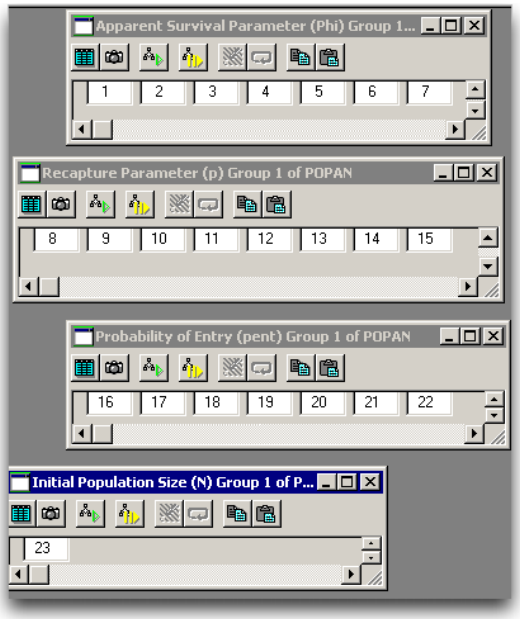
Let us begin by fitting a model **only** to the regular adults, i.e., to the first group. Later models will be fit to both groups. Launch **MARK**. Select the *POPAN* module, enter the number of sampling occasions and the number of attribute groups:



Don't forget to set the intervals between sampling occasions. As weeks (1 and 2) were pooled and labeled as week 1.5, the interval between the first and second sampling occasion is 1.5 weeks. Similarly weeks (9 and 10) were pooled and so the last interval is also longer.

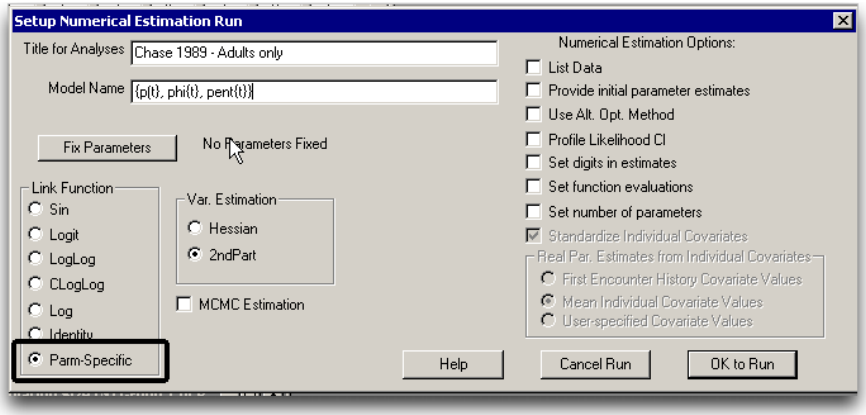


Start by fitting a fully-time dependent model $\{p_t, \varphi_t, b_t\}$ (using an obvious notation extension from CJS models). In this model there are 8 sampling occasions which give rise to 8 capture probabilities, 7 apparent survival probabilities, 8 probability of entry probabilities, and 1 super-population parameter. Note that **MARK** does *not* allow the user to specify the parameter b_0 (the proportion of the population available just before the first sampling occasion) and so only presents 7 *PENT* parameters in the PIM and output corresponding to $b_1, \ldots, b_7$.*



Note that *unlike* the CJS models, *we assume a single set of survival and catchability parameters*, regardless of when previously captured. The super-population size has its own parameter.

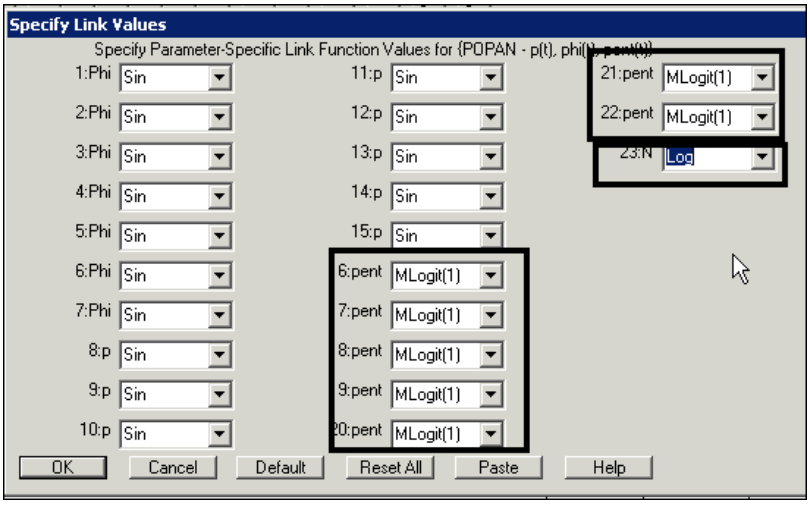
Because $b_0 + b_1 + \cdots + b_{K-1} = 1$, a special link-function must be specified for the *PENT* parameters. This is done using the 'Parameter specific link function' radio button:



* In the stand alone *POPAN* package, the user has access to all of the *PENT*'s.

The *sin* or *logit* or any of the other link functions can be used for the p and φ parameters. In order to specify that a set of parameters must sum to 1, the *Multinomial Logit* link function (called `Mlogit` in **MARK**) must be used. If there are several groups, each set of *PENT*s must independently sum to 1, so **MARK** provides several sets of `MLogit` link functions. As there is only one group, the `MLogit(1)` link-function is used for the *PENT*s. It is possible to specify that some of the *PENT*s are zero if, for example, the experimenter knew that no new animals entered the study population during this interval.

Also notice, that a *log* or *identity* link should be used for the super-population size as it is not restricted to lie between 0 and 1:



Now run **MARK**. If we look at the *REAL* estimates, we must keep in mind that not all parameters are identifiable:

Chase 1989 Adults Only

Real Function Parameters of {p(t), phi(t

Parameter	Estimate	Standard Error
1:Phi	0.5717376	0.0977208
2:Phi	0.9999994	0.6125507E-03
3:Phi	0.7191576	0.1175738
4:Phi	1.0000000	0.0000000
5:Phi	0.7826687	0.4727436
6:Phi	0.2822160	0.2337353
7:Phi	0.9975004	2.0560912
8:p	0.9999823	0.0133128
9:p	0.3751059	0.1375089
10:p	0.2346356	0.0392107
11:p	0.3697123	0.0623328
12:p	0.3077557	0.0556362
13:p	0.2146932	0.1361140
14:p	0.2651052	0.1474306
15:p	0.1303810	0.4117473
16:pent	0.1360109	0.0612740
17:pent	0.5259542	0.0936811
18:pent	0.1648277E-11	0.3260178E-12
19:pent	0.1648277E-11	0.3260178E-12
20:pent	0.0631742	0.0777153
21:pent	0.1553686	0.0804563
22:pent	0.1648277E-11	0.3260178E-12
23:N	311.75769	38.375642

In particular, the final survival and catchability are confounded as in the CJS model. The initial entrance and catchability parameters are also confounded (only the product $b_0 p_1$ can be estimated) – however, **MARK** does **not** report b_0 so the first *PENT* reported here refers to b_1 which cannot be estimated separately (refer to Table 12.2). This non-identifiability can often be recognized by the large standard errors for certain estimates, or by estimates tending to the value 1.0. Also notice, that because the intervals are unequal size, the survival probabilities are given on a *per week basis*, so that the survival probability for the initial 1.5 week interval is found as $0.57^{1.5} = 0.43$.

The estimates of population size and net births are found under the derived parameter section:

Grp.	Occ.	B-hat	Standard Error
1	1	42.654354	10.772958
1	2	163.97026	29.665562
1	3	0.5138632E-09	0.7960027E-10
1	4	0.5138632E-09	0.7960027E-10
1	5	19.695050	25.021000
1	6	48.437357	28.200096
1	7	0.5138632E-09	0.7960027E-10

Population Estimates of {p(t),

Grp.	Occ.	N-hat	Standard Error
1	1	37.000665	5.7316095
1	2	58.650105	20.302137
1	3	221.62033	26.690452
1	4	151.46921	21.665511
1	5	149.46921	21.665511
1	6	130.41856	80.203341
1	7	82.985839	44.256052
1	8	76.698288	241.27819

Again, not all parameters are identifiable. The derived parameters also include estimates of gross births - these are explained in more detail in Schwarz *et al*. (1993).

Goodness-of-fit can be assessed using the **RELEASE** suite as in CJS models (see Chapter 5 for details on **RELEASE**). The results are shown below.

Summary of TEST 3 (Goodness of fit) Results

Group	Component	Chi-square	df	P-level	Sufficient Data
1	3.SR2	0.2652	1	0.6065	No
1	3.SR3	0.0000	1	1.0000	No
1	3.SR4	0.0903	1	0.7638	Yes
1	3.SR5	0.0000	1	1.0000	No
1	3.SR6	0.3434	1	0.5579	No
1	3.SR7	0.0000	0	1.0000	No
Group 1	3.SR	0.6989	5	0.9830	
1	3.Sm2	0.0000	1	1.0000	No
1	3.Sm3	0.0000	1	1.0000	No
1	3.Sm4	0.1848	1	0.6673	Yes
1	3.Sm5	0.0000	1	1.0000	No
1	3.Sm6	0.0000	0	1.0000	No
Group 1	3.Sm	0.1848	4	0.9960	
Group 1	TEST 3	0.8837	9	0.9997	

Summary of TEST 2 (Goodness of fit) Results

Group	Component	Chi-square	df	P-level	Sufficient Data
1	2.C2	0.1942	1	0.6594	Yes
1	2.C3	6.1651	2	0.0458	Yes
1	2.C4	0.4521	1	0.5013	Yes
1	2.C5	0.2399	1	0.6244	Yes
1	2.C6	2.5951	1	0.1071	No
Group 1	TEST 2	9.6463	6	0.1404	

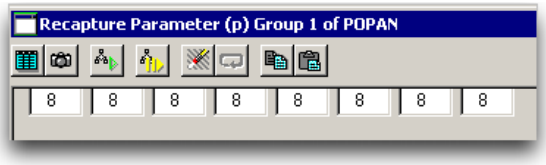
Goodness of Fit Results (TEST 2 + TEST 3) by Group

Group	Chi-square	df	P-level
1	10.5300	15	0.7851

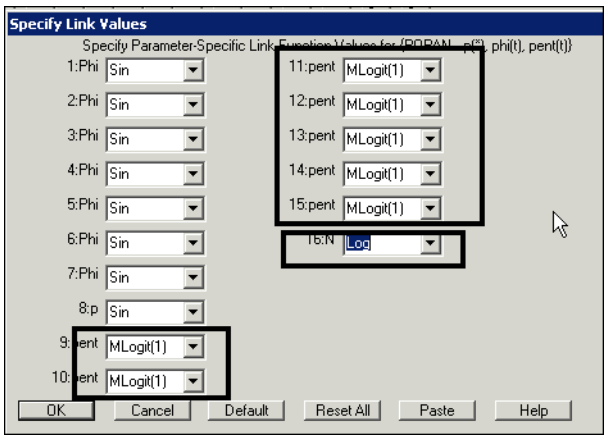
There is some evidence of potential lack-of-fit as indicated by **Test2** in component **C3**, but a detailed investigation of that table shows it is not serious. Unfortunately, because individual capture histories were used, residual plots are not useful.

Because of parameter confounding, it is important to count the actual number of estimable parameters. On the surface there are 24 parameters composed of 8 capture parameters, 7 survival parameters, 8 *PENT* parameters and 1 super-population parameter. However, the *PENT*s must sum to 1, and Table 12.2 indicates that two parameters are lost to confounding which leaves a net of 21 actual identifiable parameters. Check that the results browser shows 21 parameters for this model.

The original sampling experiment has approximately equal effort at all sampling occasions. Perhaps a model with constant catchability over time is suitable, i.e., model $\{p_\bullet, \varphi_t, b_t\}$. This model is specified in the PIM in the usual fashion:

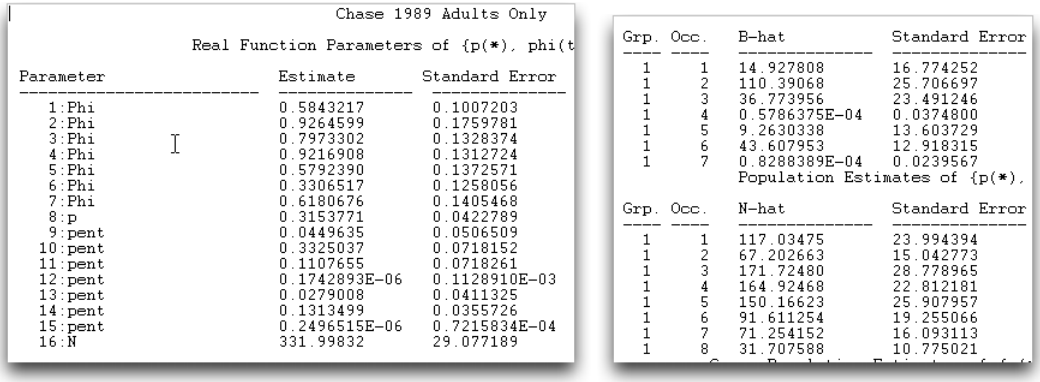


None of the other PIM's need to be respecified. The model is run, and again the parameter specific link functions must be specified for the *PENT*s (use the *MLogit(1)* link function) and for the super-population size (use the log link function):



Now all parameters are identifiable (shown in the following two figures at the top of the next page). The results show $\hat{b}_1 = 0.044$, $\hat{b}_2 = 0.33$, etc. These are interpreted as 4.4% of adult returning salmon return between weeks 1.5 and 3; about 33% of adult returning salmon return to spawn between weeks 4 and 5, etc. The value of $\hat{b}_0 = 0.352$ is obtained by subtraction ($\hat{b}_0 = 1 - \hat{b}_1 - \hat{b}_2 - \cdots - \hat{b}_{K-1}$). This is interpreted as 35% of adults returning salmon has returned to spawn before sampling began in week 1.5.

The total number of salmon returning to spawn (the super-population) is estimated to be $\hat{N} = 332$ (SE=29) fish. The derived birth parameters are found as $\hat{B}_i = \hat{N}\hat{b}_i$. For example, $\hat{B}_1 = \hat{N}\hat{b}_1 = (332 \times 0.044) = 14.9$. This is interpreted as about 15 adult fish returned to spawn between weeks 1.5 and 3. $\hat{B}_0 = \hat{N}\hat{b}_0 = (332 \times 0.352) = 117$ fish are estimated to be present before the first sampling occasion.



Chase 1989 Adults Only

Real Function Parameters of {p(*), phi(t

Parameter	Estimate	Standard Error
1:Phi	0.5843217	0.1007203
2:Phi	0.9264599	0.1759781
3:Phi	0.7973302	0.1328374
4:Phi	0.9216908	0.1312724
5:Phi	0.5792390	0.1372571
6:Phi	0.3306517	0.1258056
7:Phi	0.6180676	0.1405468
8:p	0.3153771	0.0422789
9:pent	0.0449635	0.0506509
10:pent	0.3325037	0.0718152
11:pent	0.1107655	0.0718261
12:pent	0.1742893E-06	0.1128910E-03
13:pent	0.0279008	0.0411325
14:pent	0.1313499	0.0355726
15:pent	0.2496515E-06	0.7215834E-04
16:N	331.99832	29.077189

Grp.	Occ.	B-hat	Standard Error
1	1	14.927808	16.774252
1	2	110.39068	25.706697
1	3	36.773956	23.491246
1	4	0.5786375E-04	0.0374800
1	5	9.2630338	13.603729
1	6	43.607953	12.918315
1	7	0.8288389E-04	0.0239567

Population Estimates of {p(*),

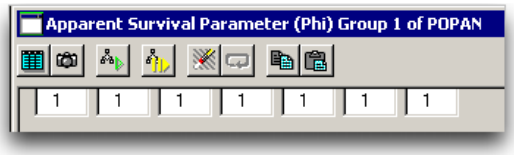
Grp.	Occ.	N-hat	Standard Error
1	1	117.03475	23.994394
1	2	67.202663	15.042773
1	3	171.72480	28.778965
1	4	164.92468	22.812181
1	5	150.16623	25.907957
1	6	91.611254	19.255066
1	7	71.254152	16.093113
1	8	31.707588	10.775021

The derived estimates of population size are found iteratively and must account for losses-on-capture and the unequal time intervals (which affects the survival terms).

$$\hat{N}_1 = \hat{B}_0 = 117.03.$$
$$\hat{N}_2 = (\hat{N}_1 - loss_1)\hat{\varphi}_1^{1.5} + \hat{B}_1 = (117.03 - 0)0.584^{1.5} + 14.92 = 67.2.$$
$$\hat{N}_3 = (\hat{N}_2 - loss_2)\hat{\varphi}_2^{1.0} + \hat{B}_2 = (67.2 - 1)0.926^{1.0} + 110.39 = 171.72$$
$$\vdots$$

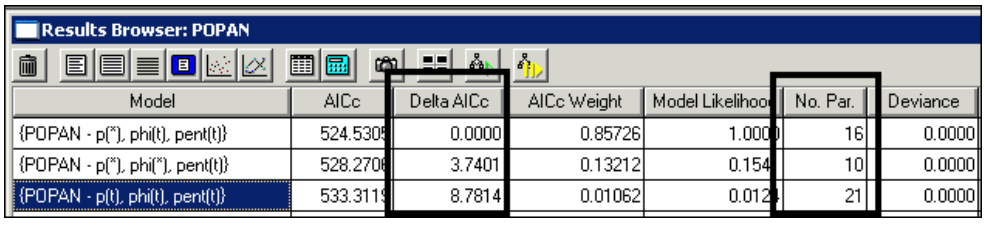
The number of identifiable parameters for this model is 16 composed of 1 capture parameters, 7 survival parameters, 8 *PENT* parameters and 1 super-population parameter less the restriction that the *PENT*s must sum to 1. The number of parameters reported in the browser may have to be manually adjusted to indicate the correct number of parameters.

Another sub-model can also be fit where the apparent survival probability (per unit time) is constant over all intervals, i.e., model $\{p_\bullet, \varphi_\bullet, b_t\}$. It is fit in the same fashion by adjusting the PIMs:



This model would have 10 parameters composed of 1 capture parameter, 1 survival parameter, 8 *PENT* parameters, and 1 super-population parameters with 1 restriction that the *PENT*s sum to 1.

The final results table (after making sure that the number of parameters is correct):



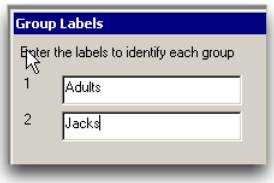
Model	AICc	Delta AICc	AICc Weight	Model Likelihood	No. Par.	Deviance
{POPAN - p(*), phi(t), pent(t)}	524.5305	0.0000	0.85726	1.0000	16	0.0000
{POPAN - p(*), phi(*), pent(t)}	528.2706	3.7401	0.13212	0.1541	10	0.0000
{POPAN - p(t), phi(t), pent(t)}	533.3119	8.7814	0.01062	0.0124	21	0.0000

shows not much support for this final model. If model averaging is to be used, some care must be taken as not all parameters are identifiable in all models. For example, most models with a p_t structure, will be unable to estimate abundance at the first (N_1) or last (N_K) sampling occasion; nor can recruitment be estimated for the first (b_1, or B_1) or last (b_{K-1} or B_{K-1}) interval.

Other models could be fit, e.g., an equal fraction of the super-population returns to spawn in each week, but in this example is highly unlikely biologically and was not fit.

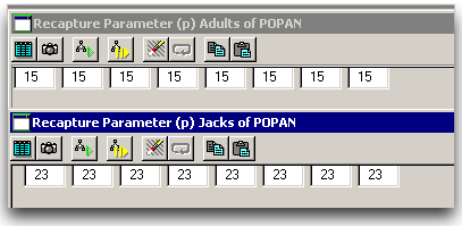
Now let us return to the real question of interest - do jacks and adults have the same return pattern? This is now a two group problem and is handled in a similar fashion to the ordinary CJS model.

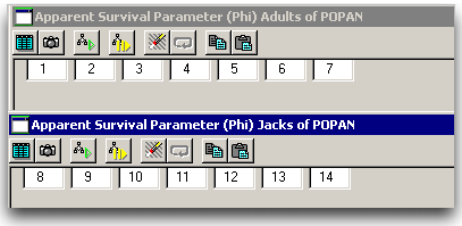
Start a new project and this time specify **two** groups (regular adults and jacks) rather than a single group. Also specify the names of the two groups.

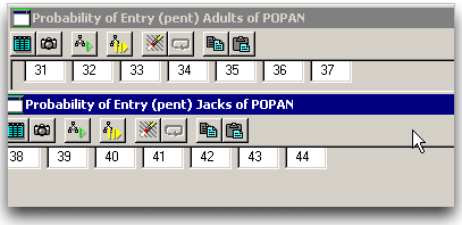


Now each parameter can vary over sampling occasions (intervals) and/or groups. The full model fit will be specified by a triplet of specifications.

In the previous example, a model with equal catchability over sampling occasions was tenable, but adults and jacks may have different catchabilities. Survival probabilities varied by time, so perhaps start with a fully time- and group-dependent model for φ. Also start with a full group and time dependence for the *PENT*s. This would correspond to model $\{p_g, \varphi_{g*t}, b_{g*t}\}$ with the following PIMs.

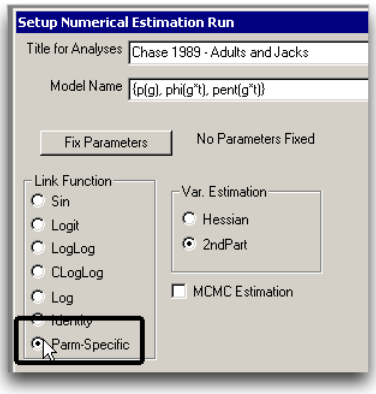




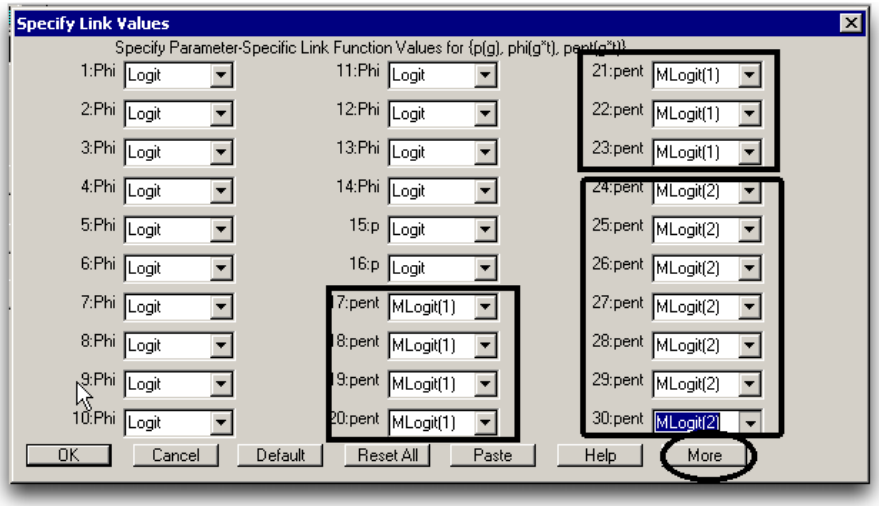


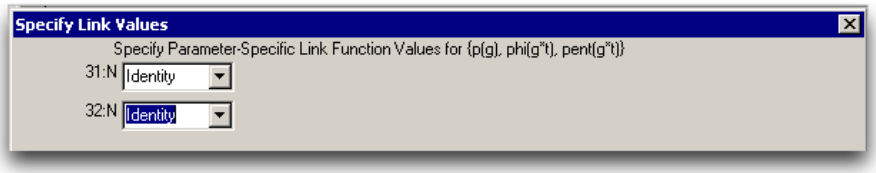
The two PIMs for the super-population size of the adults and jacks (respectively) are not shown.

Request that **MARK** fit this model. As before, we must indicate to **MARK** that the *PENT*s sum to 1, for each group using the parameter specific link functions:.



There are a total of 8 sampling occasions, 8 *PENT*s per group, but **MARK** only shows the last seven (b_0 for each group is implicitly assumed). Use the `MLogit(1)` link function for the first seven *PENT*s (belonging to group 1) and the `MLogit(2)` link function for the last seven *PENT*s (belonging to group 2). Notice that only 30 parameters are displayed per window, so the '**`More`**' button must be used to scroll to the next page:





Don't forget to specify that the two super-population parameters should have the identity or log link function.

Run the model and add it to the results browser. Then, run and fit the models $\{p_g, \varphi_t, b_{g*t}\}$ and and $\{p_g, \varphi_t, b_t\}$.*

Count parameters carefully. All of the models have a simple structure for the p's so there is no problem with confounding. The model $\{p_g, \varphi_t, b_{g*t}\}$ has 2 catchability parameters, 7 survival parameters, 16 *PENT*s, and 2 super-population sizes. However, each set of *PENT*s must sum to 1, leaving 25 free parameters. The model $\{p_g, \varphi_t, b_t\}$ has 2 catchability parameters, 7 survival parameters, 8 PENTs (but these must sum to 1), and 2 super-population parameters for a total of 18 parameters. The model $\{p_g, \varphi_{g*t}, b_{g*t}\}$ has 2 catchability parameters, 14 survival parameters, 18 *PENT*s (but each of the two groups of *PENT*s must sum to 1), and 2 super-population sizes for a total of 32 parameters.

The final results window looks like (after adjusting for the number of parameters)

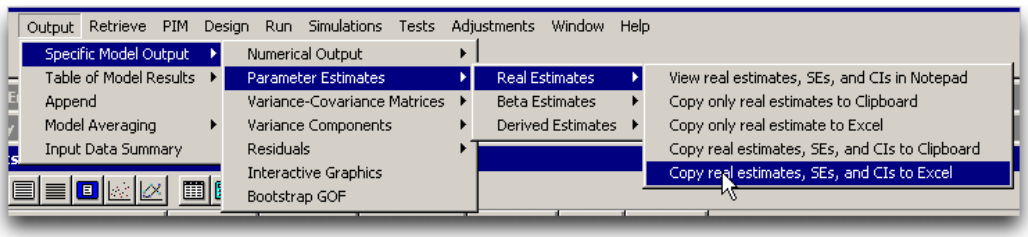
Results Browser: POPAN

Model	AICc	Delta AICc	AICc Weight	Model Likelihood	No. Par.	Deviance
{POPAN 2 groups ({p(g), phi(t), pent(g*t)}	933.4122	0.0000	0.95057	1.0000	25	0.0000
{POPAN 2 groups ({p(g), phi(t), pent(t)}	940.5470	7.1348	0.02683	0.0282	18	0.0000
{POPAN 2 groups ({p(g), phi(g*t), pent(g*t)}	940.8910	7.4788	0.02259	0.0238	32	0.0000

The final models of interest are a comparison between model $\{p_g, \varphi_t, b_{g*t}\}$ and $\{p_g, \varphi_t, b_t\}$ (why?). The ΔAIC_c shows very little support for the model where the jacks enter in the same distribution as the adults. In particular, examine the estimates of the *PENT*s for the $\{p_g, \varphi_t, b_{g*t}\}$ model:

10:pent	0.0333328	0.0457877
11:pent	0.3268064	0.0685035
12:pent	0.1307572	0.0752736
13:pent	0.0011528	0.0591666
14:pent	0.0410032	0.0386589
15:pent	0.1214374	0.0324185
16:pent	0.1719801E-08	0.0000000
17:pent	0.9631933E-11	0.1043234E-11
18:pent	0.2276932	0.0835763
19:pent	0.0362517	0.0864785
20:pent	0.1635956E-09	0.0000000
21:pent	0.0056390	0.0455758
22:pent	0.0368692	0.0266959
23:pent	0.5456831E-06	0.4850133E-03

Recall that even though there are 8 *PENT* parameters per group that **MARK** does not let you specify the b_0 parameter in the PIMS – this value is obtained by subtraction from 1. Further manipulations can be done by copying the real parameter values to an Excel spreadsheet:



* The *Initial* → *Copy 1 PIM to another PIM* is helpful here.

You will find that about 34% (1 − 0.033 − 0.326 − . . . − 0.000) of adults were in the stream prior to the first sampling occasion, while about 69% (1 − 0.000 − 0.227 − 0.036 − . . . − 0.000) of jacks were in the stream prior to the first sampling occasion. So it appears that precocious males tend to return earlier (for this stream and year) than regular adults.

In this example, it is vitally important that catchability be approximately constant across all sampling occasions so that a model with p_g could be fit; any model where catchability varied across time (a p_t or p_{g*t} model) would have the first *PENT* parameter hopelessly confounded with the first catchability parameter and in many cases, makes it difficult if not impossible to do sensible model comparisons. This again illustrates the need for careful study design with JS models.

begin sidebar

The multinomial logit link and the POPAN model

In situations where you want to constrain estimates from a set of 2 or more parameters to sum to 1, you might use the multinomial logit link (MLogit), which was introduced in some detail in Chapter 10 with respect to multi-state models (where the transitions from a given stratum must logically sum to 1.0).

While specifying the MLogit link in **MARK** is straightforward, you need to be somewhat careful. Consider the following set of parameters in a PIM for the probability of entry (*pent*) in a **POPAN** model:

```
61 62 63 64 65 66 67 68 69 70
```

The parameter-specific link would be selected in the 'Setup Numerical Estimation Run' window, and the MLogit(1) link would be applied to parameters 61 → 70 to force these 10 estimates to sum to ≤ 1. But suppose that you wanted to force *all* of the 10 entry probabilities to be the same, and have the sum of all 10 be ≤ 1? You might be tempted to specify a PIM such as

```
61 61 61 61 61 61 61 61 61 61
```

(i.e., simply use the same index value for all the parameters in the PIM), but that would be incorrect. Changing the PIM and selecting the MLogit link for parameter 61 would result in parameter 61 alone summing to ≤ 1 (i.e., just like a logit link), but would not force the sum of the 10 values of parameter 61 to sum to ≤ 1.

To implement the proposed model, the PIM should not be changed from the top example (i.e., it should maintain the indexing from 61 → 70), and the design matrix should be used to force the same estimate for parameters 61 → 70:

```
Parameter    Design Matrix
   61                1
   62                1
   63                1
   64                1
   65                1
   66                1
   67                1
   68                1
   69                1
   70                1
```

Then the MLogit(1) link should be specified for the 10 parameters 61 → 70. The result is that now all 10 parameters have the same value, and 10 times this value is $le1$.

Another example - suppose you wanted parameters 61 and 62 to be the same value, 63 to 66 the same, 67 and 68 the same, and 69 and 70 the same, but the sum over all parameters to be ≤ 1. Again you would use the PIM

```
61 62 63 64 65 66 67 68 69 70
```

but again use the design matrix to implement the constraints. The following design matrix is one example that would produce such a set of constraints.

```
Parameter Design Matrix
    61          1 1 0 0
    62          1 1 0 0
    63          1 0 1 0
    64          1 0 1 0
    65          1 0 1 0
    66          1 0 1 0
    67          1 0 0 1
    68          1 0 0 1
    69          1 0 0 0
    70          1 0 0 0
```

The key point with these examples is that the PIM *cannot* be used to constrain parameters if you want the entire set of parameters to sum to ≤ 1. Rather, the design matrix has to be used to make the constraints, with each of the entries in the PIM given the same MLogit(x) link. Further examples of the MLogit link are discussed in Chapter 10.

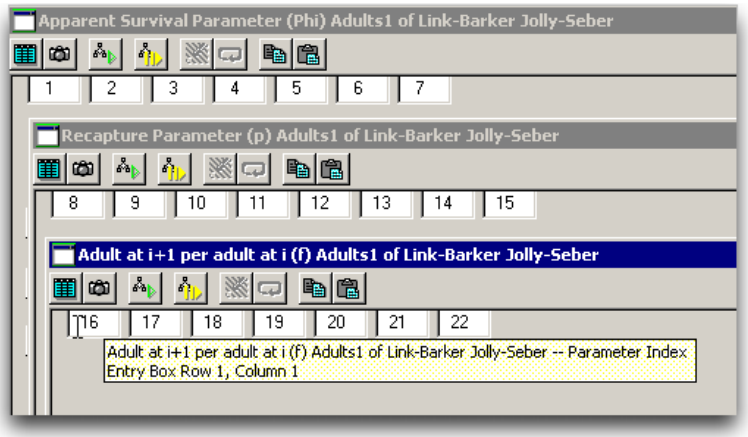
end sidebar

12.4.2. Link-Barker and Pradel-recruitment formulations

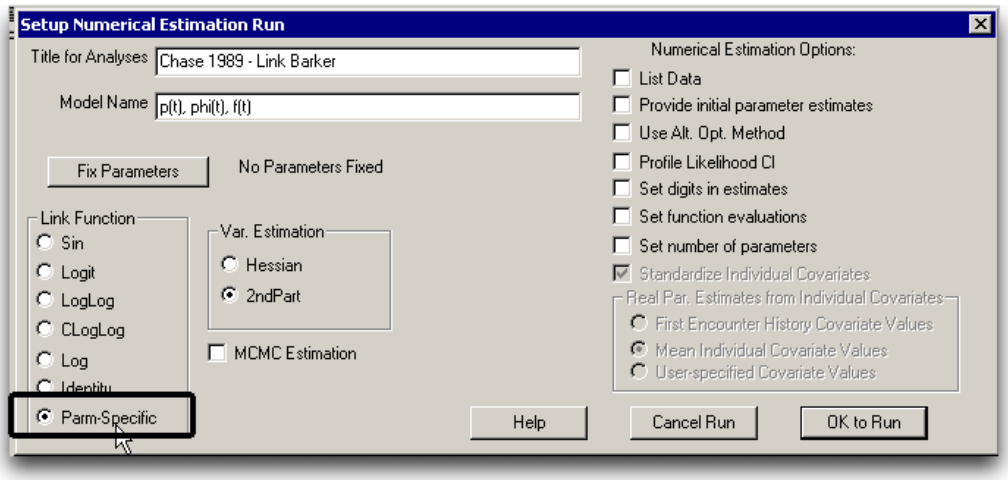
The Link-Barker or Pradel-recruitment formulation can be conveniently obtained by switching data types from any of the JS formulations. We will illustrate the use of the Link-Barker formulation; that for the Pradel-recruitment is similar, but in this case cannot be used because of losses-on-capture.

Let us begin with a time dependent model for all parameters, i.e., $\{p_t, \varphi_t, f_t\}$. The number of groups and sampling intervals would have been entered as seen in the *POPAN* formulation. The survival and catchability PIMs mimic those for the *POPAN* formulation.

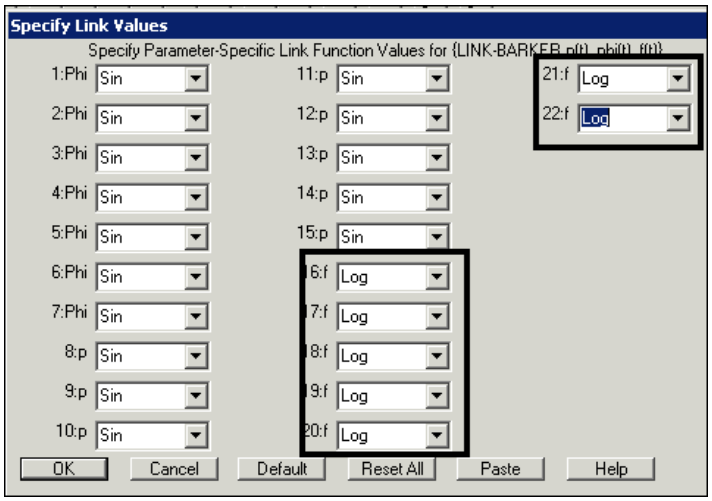
In the Link-Barker formulation, there are $K - 1 = 7$ population recruitment parameters with a standard PIM:



Because the population recruitment value is **not** limited to lie between 0 and 1 (for example, the recruitment value could exceed 1), the '`Parameter Specific Link`' functions should be specified when models are run:*



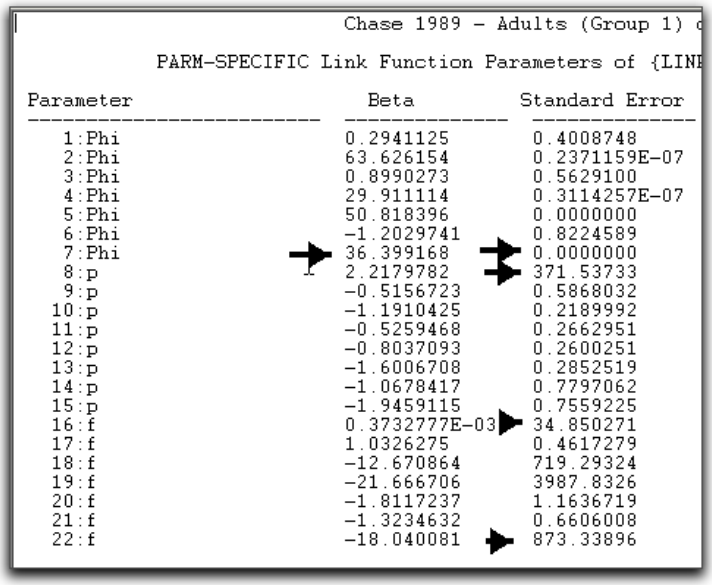
Any of the link functions can be used for the catchability and survival parameters (although the *logit* and *sin* link are most common), but either the *log* or the *identity* link function should be used for the population recruitment parameters. There are no restrictions that the recruitment parameters sum to 1 over experiment so the `Mlogit` link should **not** be used.



Run the model and append it to the browser.

As in the *POPAN* formulation, the fully time-dependent Link-Barker and Pradel-recruitment formulations suffer from confounding. If you examine the β parameter estimates, and the estimated SE (shown at the top of the next page), there are several clues that confounding has taken place:

* An undocumented feature of **MARK** is that it will use the *log* link for the recruitment parameter if you specify a *logit* or *sin* link in the radio buttons.

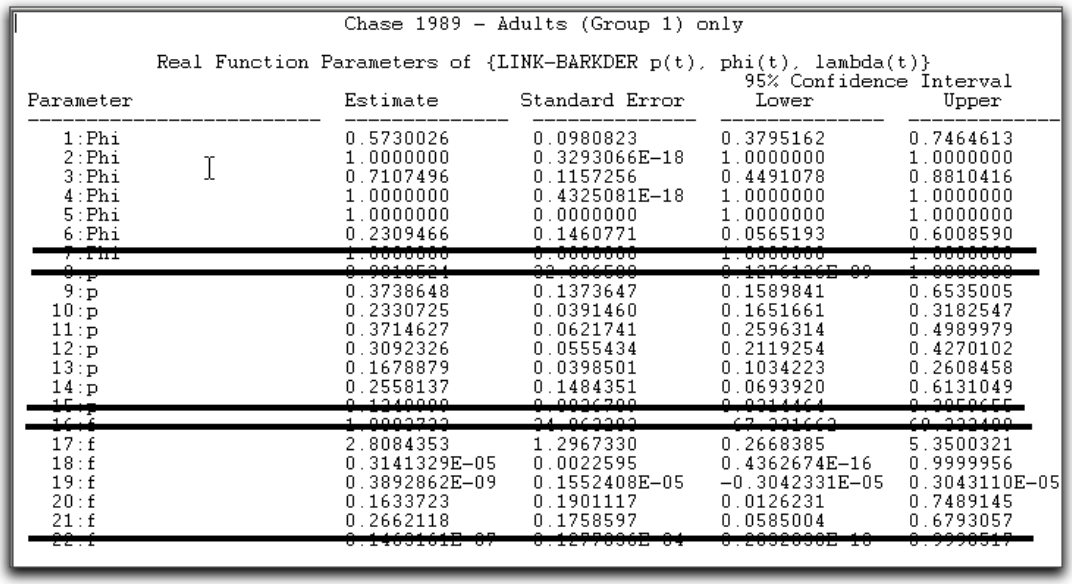


Chase 1989 - Adults (Group 1) c

PARM-SPECIFIC Link Function Parameters of {LINI

Parameter	Beta	Standard Error
1:Phi	0.2941125	0.4008748
2:Phi	63.626154	0.2371159E-07
3:Phi	0.8990273	0.5629100
4:Phi	29.911114	0.3114257E-07
5:Phi	50.818396	0.0000000
6:Phi	-1.2029741	0.8224589
7:Phi	36.399168	0.0000000
8:p	2.2179782	371.53733
9:p	-0.5156723	0.5868032
10:p	-1.1910425	0.2189992
11:p	-0.5259468	0.2662951
12:p	-0.8037093	0.2600251
13:p	-1.6006708	0.2852519
14:p	-1.0678417	0.7797062
15:p	-1.9459115	0.7559225
16:f	0.3732777E-03	34.850271
17:f	1.0326275	0.4617279
18:f	-12.670864	719.29324
19:f	-21.666706	3987.8326
20:f	-1.8117237	1.1636719
21:f	-1.3234632	0.6606008
22:f	-18.040081	873.33896

As in the *POPAN* formulation, survival in the last interval and catchability at the last sampling occasion are confounded. The corresponding β parameters are very large on the logit scale with standard errors that are either zero or very large. Similarly, p_1 cannot be estimated and its β standard error is very large. Finally, because the first and last catchabilities cannot be estimated, neither can 'population size' (despite population size not being explicitly in the model) and so the recruitment parameter (the f_i values) based on occasion 1 (f_1) or terminating with occasion K (f_{K-1}) cannot be estimated either. We notice that the standard errors for these recruitment parameters are nonsensical.



Chase 1989 - Adults (Group 1) only

Real Function Parameters of {LINK-BARKDER p(t), phi(t), lambda(t)}

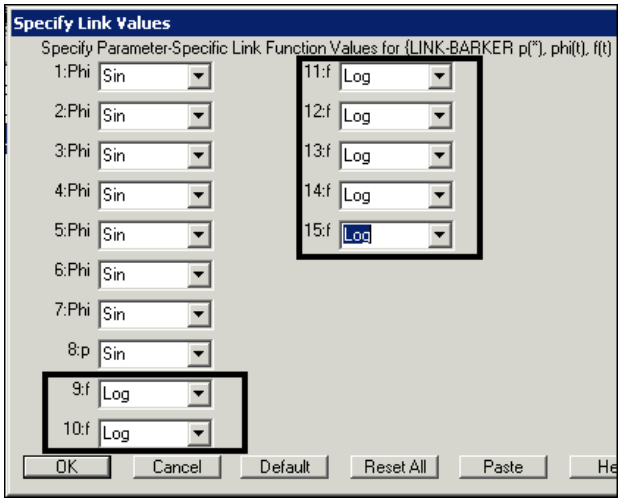
Parameter	Estimate	Standard Error	95% Confidence Interval Lower	Upper
1:Phi	0.5730026	0.0980823	0.3795162	0.7464613
2:Phi	1.0000000	0.3293066E-18	1.0000000	1.0000000
3:Phi	0.7107496	0.1157256	0.4491078	0.8810416
4:Phi	1.0000000	0.4325081E-18	1.0000000	1.0000000
5:Phi	1.0000000	0.0000000	1.0000000	1.0000000
6:Phi	0.2309466	0.1460771	0.0565193	0.6008590
~~7:Phi~~	~~1.0000000~~	~~0.0000000~~	~~1.0000000~~	~~1.0000000~~
~~8:p~~	~~0.9818524~~	~~32.886588~~	~~0.1276126E-09~~	~~1.0000000~~
9:p	0.3738648	0.1373647	0.1589841	0.6535005
10:p	0.2330725	0.0391460	0.1651661	0.3182547
11:p	0.3714627	0.0621741	0.2596314	0.4989979
12:p	0.3092326	0.0555434	0.2119254	0.4270102
13:p	0.1678879	0.0398501	0.1034223	0.2608458
14:p	0.2558137	0.1484351	0.0693920	0.6131049
~~15:p~~	~~0.1240000~~	~~0.0026700~~	~~0.0314464~~	~~0.3050655~~
~~16:f~~	~~1.0003733~~	~~34.063303~~	~~67.331662~~	~~69.332400~~
17:f	2.8084353	1.2967330	0.2668385	5.3500321
18:f	0.3141329E-05	0.0022595	0.4362674E-16	0.9999956
19:f	0.3892862E-09	0.1552408E-05	-0.3042331E-05	0.3043110E-05
20:f	0.1633723	0.1901117	0.0126231	0.7489145
21:f	0.2662118	0.1758597	0.0585004	0.6793057
~~22:f~~	~~0.1463161E-07~~	~~0.1277836E-04~~	~~0.2832838E-18~~	~~0.9998517~~

The user must be *very* careful to count parameters carefully and to see if **MARK** has detected the correct number of parameters. There are 8 sampling occasions. The fully time-dependent model has, on

the surface, 8 capture parameters, 7 survival parameters, and 7 recruitment recruitment parameters for a total of 22 parameters. However, as shown in Table 12.3, there are two parameters lost to confounding which gives a total of 20 parameters that can be estimated. The number of parameters reported in the results browser may have to be modified manually.

As in the *POPAN* formulation, models where constraints are placed on the initial and final catchabilities can resolve this confounding. Because roughly the same effort was used in all sampling occasions, the model $\{p_\bullet, \varphi_t, f_t\}$ seems appropriate.

Adjust the PIM for the recapture probabilities to be constant over time, re-run the model (don't forget to use the *Parameter Specific Link* functions),*



and append the results to the browser. Let's look at the parameter estimates:

Chase 1989 - Adults (Group 1) only

Real Function Parameters of {LINK-BARKER p(*), phi(t), f(t) - indiv sin link}

Parameter	Estimate	Standard Error	95% Confidence Interval Lower	Upper
1:Phi	0.5833992	0.1006706	0.3834096	0.7592529
2:Phi	0.9279969	0.1765803	0.0676519	0.9995634
3:Phi	0.7970321	0.1330251	0.4393349	0.9516419
4:Phi	0.9221987	0.1316963	0.2450770	0.9976947
5:Phi	0.5800559	0.1375482	0.3135128	0.8068629
6:Phi	0.3308433	0.1259074	0.1395454	0.6011657
7:Phi	0.6190726	0.1399812	0.3367479	0.8387623
8:p	0.3137171	0.0420822	0.2375970	0.4013852
9:f	-9.2521580	0.2045257	0.0386820	0.7385912
10:f	1.6441301	0.5502536	0.5656329	2.7226273
11:f	0.2114983	0.1593520	0.0395904	0.6357435
12:f	0.2728460E-07	0.6223856E-04	0.3789270E-18	0.9994913
13:f	0.0571923	0.0884807	0.0024271	0.6019896
14:f	0.4270444	0.1464334	0.1874095	0.7066338
15:f	0.1160248E-04	0.0108513	0.1611365E-15	0.9999988

Goodness-of-fit analysis is done using the **RELEASE** suite as seen in the *POPAN* fit and as explained in Chapter 5.

The number of parameters that can be estimated is now 15 being composed of 1 capture parameter,

* *CAUTION*: When we ran this model with the *logit* link for the φ's, one φ converged to a value of 1; we would recommend that the *sin* link be used.

7 survival parameters, and 7 recruitment parameters.

If you compare the estimates of p and φ between the Link-Barker formulation and the *POPAN* formulation they are identical (except for rounding errors). Note that because of unequal time intervals, estimates of φ_i are on a per-unit basis. The actual survival in the first and last interval must be obtained by raising the reported φ's to the 1.5^{th} power (corresponding to the 1.5 week interval).

The *PENT*s from *POPAN* and the f's from the Link-Barker are not directly comparable. However, the following equivalents are noted:

$$(\hat{f}_1^{LB})^{1.5} = 0.252^{1.5} = 0.127 = \frac{\hat{B}_1^{POPAN}}{\hat{N}_1^{POPAN}} = \frac{14.92}{117.034}$$

$$\hat{f}_2^{LB} = 1.644 = \frac{\hat{B}_2^{POPAN}}{\hat{N}_2^{POPAN}} = \frac{110.39}{67.20}$$

$$\hat{f}_3^{LB} = 0.211 = \frac{\hat{B}_3^{POPAN}}{\hat{N}_3^{POPAN}} = \frac{36.77}{171.72}$$

$$\ldots$$

Similarly, estimates of population growth are also equivalent:

$$(\hat{f}_1^{LB})^{1.5} + (\hat{\varphi}_1^{LB})^{1.5} = 0.583^{1.5} + 0.252^{1.5} = 0.571 = \frac{\hat{N}_2^{POPAN}}{\hat{N}_1^{POPAN}} = \frac{67.20}{117.03} = 0.574$$

$$\hat{f}_2^{LB} + \hat{\varphi}_2^{LB} = 1.644 + 0.928 = 2.57 = \frac{\hat{N}_3^{POPAN}}{\hat{N}_2^{POPAN}} = \frac{171.72}{67.20} = 2.55$$

$$\ldots$$

If you examine the results browser (after any changes for the actual number of parameters that are estimated):

Results Browser: Link-Barker Jolly-Seber

Model	AICc	Delta AICc	AICc Weight	Model Likelihood	No. Par.	Deviance
{LINK-BARKER p(*), phi(t), f(t) }	1204.4444	0.0000	0.98096	1.0000	15	169.0603
{LINK-BARKER p(*), phi(*), f(t) - indiv sin link}	1212.8183	8.3739	0.01490	0.0152	9	186.3137
{LINK-BARKER p(t), phi(t), f(t)}	1215.3806	10.9362	0.00414	0.0042	20	166.0108

you will also see that the ΔAIC_c values between these two models matches very closely with the difference in the *POPAN* formulation. The differences in ΔAIC_c between the two formulation are artifacts of the different number of parameters estimated and the small sample correction applied to the AIC. If the actual AIC values from the two formulations are compared, the difference in AIC are nearly identical because the two formulations are simply re-parameterizations of the same models for modeling the marked animals and only differ in estimating the super-population size. As Sanathanan (1972, 1977) showed, the conditional approach of Link and Barker (2005) is asymptotically equivalent to the full likelihood approach of Schwarz and Arnason (1996). For example, in the table shown at the top of the next page, the log-likelihood and AIC (before small sample corrections are applied) were extracted from the model outputs. The differences in the log-likelihoods and the AIC among the models in different formulations are nearly the same.

POPAN formulation				*Link-Barker* formulation			
Model	$-2 \times \log \mathcal{L}$	# parms	AIC	Model	$-2 \times \log \mathcal{L}$	# parms	AIC
$\{p_\bullet, \varphi_t, b_t\}$	489.94	16	521.94	$\{p_\bullet, \varphi_t, f_t\}$	1176.74	15	1206.74
$\{p_\bullet, \varphi_\bullet, b_t\}$	507.25	10	527.25	$\{p_\bullet, \varphi_\bullet, f_t\}$	1193.99	9	1211.99
$\{p_t, \varphi_t, b_t\}$	486.80	21	528.80	$\{p_t, \varphi_t, f_t\}$	1173.69	20	1213.69

The two group case can be fit in a similar fashion as seen in the *POPAN* example. However, it is not clear exactly which models should be compared because the f_i parameters in the Link-Barker formulation depend both upon the *NET* number of new births, but also upon the population size at occasion i which depends upon the pattern of previous births and survival probabilities. Consequently, even if the same birth pattern occurred between the two groups, differences in survival probabilities could result in differences in patterns of the f's. Fortunately, it appears the survival probabilities are roughly constant between groups, so a comparison of models $\{p_g, \varphi_t, f_{g*t}\}$ vs $\{p_g, \varphi_t, f_t\}$ is a valid test of the hypothesis of equal return patterns for adults and jacks.

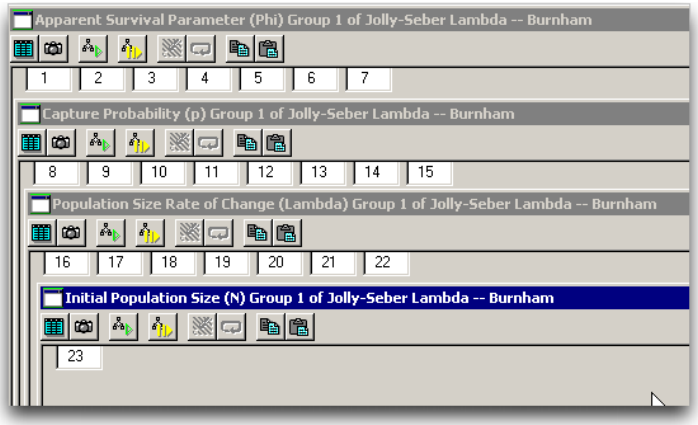
Fitting these two models to the two groups is left as an exercise for the reader.

12.4.3. Burnham Jolly-Seber and Pradel-λ formulations

The Burnham model

The Burnham JS model does not model entrants directly, but rather parameterizes changes in population size using population growth. In the case of the spawning salmon, this would correspond to the increase in the number of spawning salmon at occasion $i+1$ relative to occasion i.

The Burnham JS model is selected using the Jolly-Seber radio button. The same data file as for *POPAN* can be used. Again let us start with fitting a model just to the adults over 8 sampling occasions with unequal sampling intervals (see the screen shots in section 12.4.1 for details). Again, start by fitting the fully time-dependent model:



As in *POPAN* (and all JS formulations) there are $(K-1) = 7$ survival parameters, $K = 8$ capture

probabilities. In the Burnham model, there is a single initial population size parameter per group and $(K - 1) = 7$ population growth parameters.

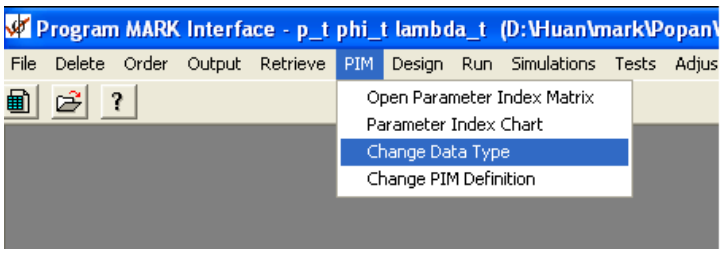
We were unable to get the Burnham Jolly-Seber model to converge for any of the models considered in this chapter. The **MARK** help files state that

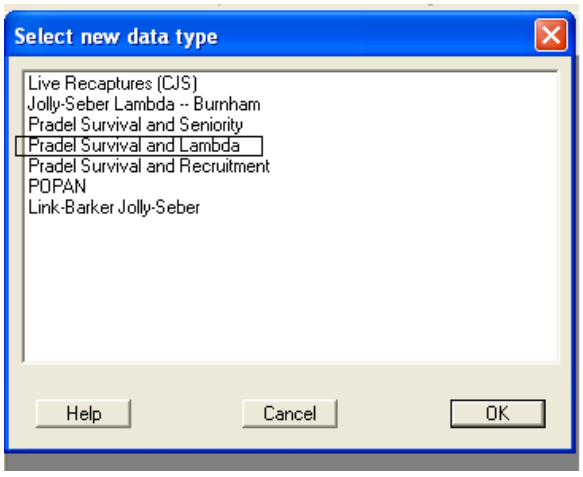
> *'This model can be difficult to get numerical convergence of the parameter estimates. Although this model has been thoroughly checked, and found to be correct, the program has difficulty obtaining numerical solutions for the parameters because of the penalty constraints required to keep the parameters consistent with each other.'*

Bummer. . . *†

The Pradel-λ model

The Pradel-λ formulation (considered in much more detail in Chapter 13) can be conveniently obtained by switching data types from any of the JS formulations (or can be entered directly from the initial screen of **MARK** as discussed in Chapter 12). The number of groups and sampling intervals would have been entered as seen in the *POPAN* formulation. Let us begin by modeling only the adult salmon.

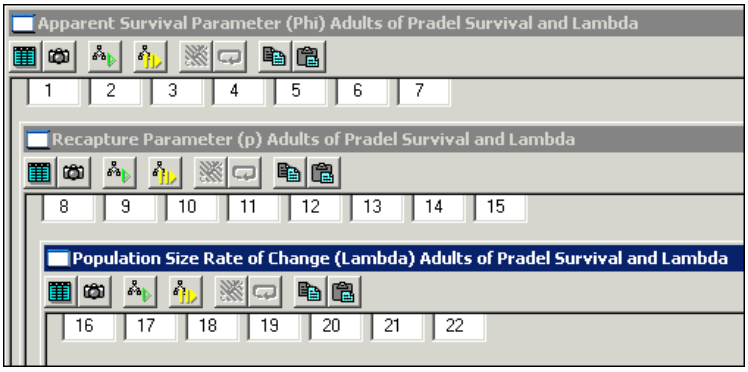




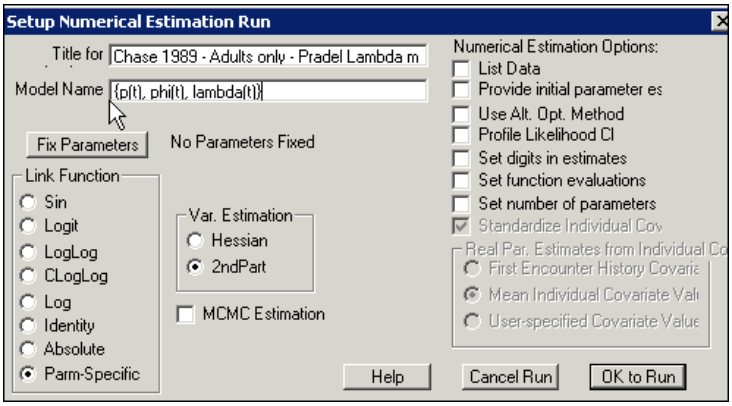
* . . . dude. C. Schwarz has clearly spent too much time on the 'wet coast' - E. G. Cooch, *pers. obs.*

† As noted earlier, this convergence problem disappears for some models if simulated annealing is used for the numerical optimization. - J. Laake & E. G. Cooch, *pers. obs.*

Begin by fitting a fully time-dependent model. The PIMs for survival and catchability mimic those seen in earlier sections; the PIM for the population growth parameter has $(K - 1) = 7$ entries:

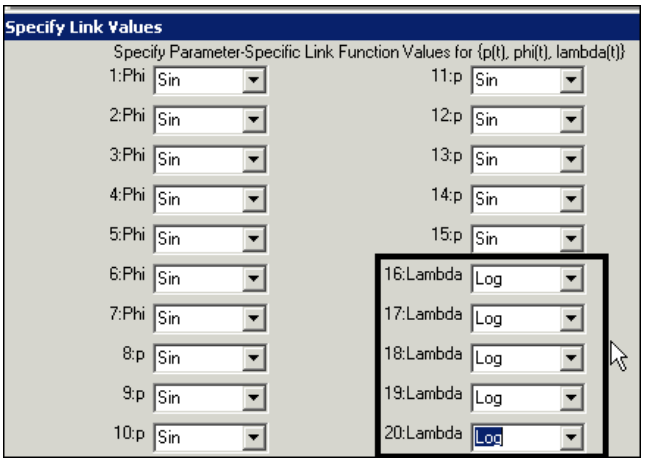


Because the population growth parameter is **not** limited to lie between 0 and 1 (for example, the growth rate could exceed 1), the *Parameter Specific Link* functions should be specified when models are run:*



Any of the link functions can be used for the catchability and survival parameters (although the *logit* and *sin* link are most common), but either the *log* or the *identity* link function should be used for the population growth parameters (this is discussed in detail in Chapter 13). There are no restrictions that the growth parameters sum to 1 over experiment so the `Mlogit` link should **not** be used. Because there are more than 20 parameters, we need to press the '**`More`**' button to specify the link function for the two remaining λ values.

* A hidden feature of **MARK** is that it will use the *log* link for the growth parameter if you specify a *logit* or *sin* link in the radio buttons.



Run the model and append it to the browser.

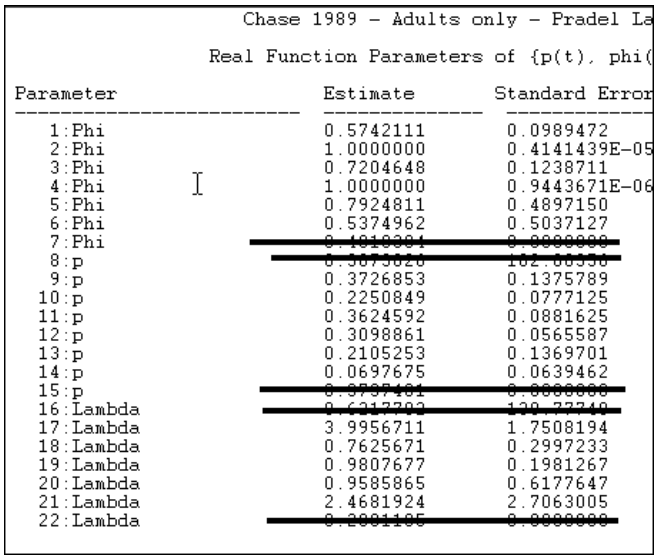
As in the *POPAN* formulation, the fully time-dependent Pradel-λ formulation suffers from confounding. If you examine the parameter estimates, there are several clues that confounding has taken place. The standard errors for some parameters are enormous or the standard error are zero. Survival in the last interval and catchability at the last sampling occasion are confounded. Similarly, λ_1 is confounded with p_1.

Chase 1989 - Adults only - Pradel La

SIN Link Function Parameters of {p(t),

Parameter	Beta	Standard Error
1:Phi	0.1489727	0.2001108
2:Phi	1.5708030	1.2446909
3:Phi	0.4566342	0.2760231
4:Phi	1.5707948	1.2177449
5:Phi	0.6248334	1.2075918
6:Phi	0.0750629	1.0102702
7:Phi	-0.0363473	0.0000000
8:p	-0.3956357	222.99293
9:p	-0.2574645	0.2845365
10:p	-0.5821609	0.1860761
11:p	-0.2786746	0.1834005
12:p	-0.3900425	0.1223032
13:p	-0.6174397	0.3359732
14:p	-1.0361815	0.2510108
15:p	-0.2552837	0.0000000
16:Lambda	-0.4751846	223.19723
17:Lambda	1.3852116	0.4381791
18:Lambda	-0.2710648	0.3930451
19:Lambda	-0.0194197	0.2020118
20:Lambda	-0.0422955	0.6444539
21:Lambda	0.9034860	1.0964706
22:Lambda	-1.5696863	0.0000000

Consequently, not all of the real parameter estimates are usable:



Chase 1989 - Adults only - Pradel La

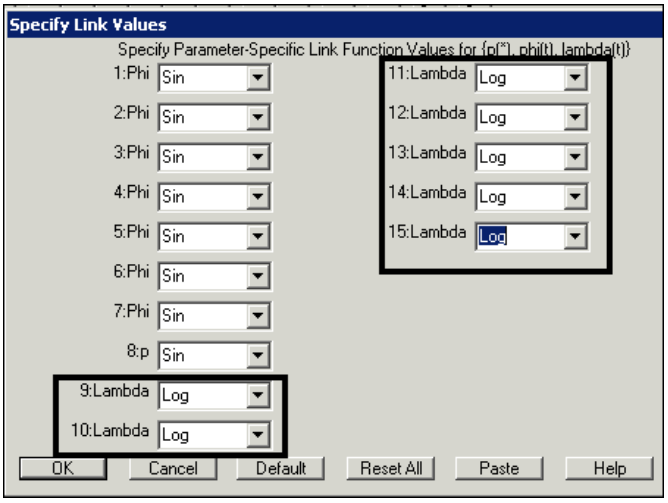
Real Function Parameters of {p(t), phi(

Parameter	Estimate	Standard Error
1:Phi	0.5742111	0.0989472
2:Phi	1.0000000	0.4141439E-05
3:Phi	0.7204648	0.1238711
4:Phi	1.0000000	0.9443671E-06
5:Phi	0.7924811	0.4897150
6:Phi	0.5374962	0.5037127
7:Phi	[illegible]	[illegible]
8:p	[illegible]	[illegible]
9:p	0.3726853	0.1375789
10:p	0.2250849	0.0777125
11:p	0.3624592	0.0881625
12:p	0.3098861	0.0565587
13:p	0.2105253	0.1369701
14:p	0.0697675	0.0639462
15:p	[illegible]	[illegible]
16:Lambda	[illegible]	[illegible]
17:Lambda	3.9956711	1.7508194
18:Lambda	0.7625671	0.2997233
19:Lambda	0.9807677	0.1981267
20:Lambda	0.9585865	0.6177647
21:Lambda	2.4681924	2.7063005
22:Lambda	[illegible]	[illegible]

The user must be very careful to count parameters carefully and to see if **MARK** has detected the correct number of parameters. There are 8 sampling occasions. The fully time-dependent model has, on the surface, 8 capture parameters, 7 survival parameters, and 7 growth parameters for a total of 22 parameters. However, there are two parameters lost to confounding which gives a total of 20 parameters that can be estimated. The number of parameters reported in the results browser may have to be modified manually.

As in the *POPAN* formulation, models where constraints are placed on the initial and final catchabilities can resolve this confounding. Because roughly the same effort was used in all sampling occasions, the model $\{p_\bullet, \varphi_t, \lambda_t\}$ seems appropriate.

Adjust the PIM for the recapture probabilities to be constant over time, re-run the model (don't forget to use the '**`Parameter Specific`** link functions or let **MARK** automatically use the log link for the λ parameters).



This gives the final estimates:

Chase 1989 - Adults only - Pradel Lamb

Real Function Parameters of {p(*), phi(t)

Parameter	Estimate	Standard Error
1:Phi	0.5927220	0.1035234
2:Phi	0.9458309	0.1859415
3:Phi	0.8060015	0.1339392
4:Phi	1.0000000	0.5549465E-06
5:Phi	0.5814686	0.1347157
6:Phi	0.2866269	0.1147553
7:Phi	0.8592311	0.2203514
8:p	0.2841406	0.0397731
9:Lambda	0.6810920	0.1085847
10:Lambda	2.6302703	0.5865236
11:Lambda	1.1418285	0.1849586
12:Lambda	0.8602454	0.1113947
13:Lambda	0.6768354	0.1320030
14:Lambda	0.8865846	0.2281068
15:Lambda	0.6158796	0.1298563

The number of parameters that can be estimated is now 15 being composed of 1 capture parameter, 7 survival parameters, and 7 λ parameters.

Note that because of unequal time intervals, estimates of φ_i are on a per-unit basis. The actual survival in the first and last interval must be obtained by raising the reported φ's to the 1.5^{th} power (corresponding to the 1.5 week interval).

Compare the estimates from the Pradel-λ formulation to those from the *POPAN* or Link-Barker formulation. Estimates of survival are similar at sampling occasions 1, 2, and 3 differing only by roundoff error.

However, the estimate of φ at sampling occasion 4 differs considerably among the models. Indeed, the Pradel-λ formulation is not even consistent as $\hat{\lambda}_4 = 0.86 < \hat{\varphi}_4 = 1.0$! The *POPAN* formulation estimated that there was **no** recruitment between sampling occasion 4 and 5 (it was constrained so that estimates of recruitment cannot be negative); the Link-Barker model also estimated the recruitment parameter to be 0 (it is also constrained so that it cannot be negative). However, there are **no** constraints in the Pradel-λ model that λ must be at least as great as survival.

The estimates of λ also don't appear to be consistent with the results from *POPAN* when estimates of population size are compared. However, this discrepancy is explained by the different ways in which losses-on-capture are incorporated into the estimates of population size and growth between the two formulations.

The model $\{p_\bullet, \varphi_\bullet, \lambda_t\}$ can also be fit and the results appended to the results browser. The estimates from this model are:

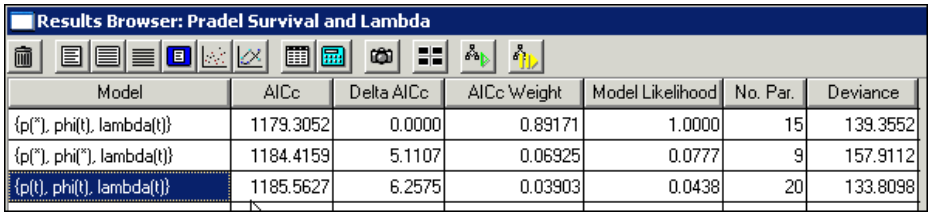
Chase 1989 - Adults only - Pradel Lambd

Real Function Parameters of {p(*), phi(*)

Parameter	Estimate	Standard Error
1:Phi	0.7139171	0.0404565
2:p	0.2979816	0.0435415
3:Lambda	0.7209730	0.1028197
4:Lambda	2.3847961	0.4978027
5:Lambda	1.1183694	0.1788025
6:Lambda	0.7287425	0.1076054
7:Lambda	0.7852352	0.1201662
8:Lambda	1.2179330	0.2103186
9:Lambda	0.5872715	0.0644242

Again, the estimates are not internally consistent as the population growth rate estimates sometimes fall below the common survival probability . These would indications that there was little or no recruitment in these intervals.

The results browser



Results Browser: Pradel Survival and Lambda

Model	AICc	Delta AICc	AICc Weight	Model Likelihood	No. Par.	Deviance
{p(*), phi(t), lambda(t)}	1179.3052	0.0000	0.89171	1.0000	15	139.3552
{p(*), phi(*), lambda(t)}	1184.4159	5.1107	0.06925	0.0777	9	157.9112
{p(t), phi(t), lambda(t)}	1185.5627	6.2575	0.03903	0.0438	20	133.8098

tells the same story as in the other formulations - strong support for the model with constant catchability and time varying survival and population growth.

The two group models can also be fit in similar fashion as in the other formulations. However, there is again the question of which models comparisons are a sensible choice. Because the λ values are population growth on a per capita basis and include both survival and recruitment, models with group- and time-dependence in λ may not be indicative of changes in recruitment between the two groups.

12.5. Example 2 - Muir's (1957) female capsid data

This second example uses the Muir (1957) data on a population of female black-kneed capsid (*Blepharidopterus angulatus*) that was originally analyzed by Jolly (1963)* and Seber (1965).

According to the British Wildlife Trust website† a black-kneed capsid is a green insect about 15 mm long that lives on orchard trees (particular apples and limes). It is a predatory insect that is beneficial to orchard owners as it feeds on red spider mites which cause damage to fruit trees. It apparently makes a 'squawk' by rubbing the tip of its beaks against its thorax and will stab people with its beak if handled.

Thirteen successive samples at alternating 3- and 4-day intervals were taken. The population is open as deaths/emigration and births/immigration can occur.

The raw history data is located in the file **capsid.inp**. A portion of the histories appears below:

```
0000000000001  47;
0000000000010  36;
0000000000011  12;
0000000000100  30;
0000000000101   8;
...
```

There is only one group (females) and the individual histories have been grouped.‡

* The data was subsequently reanalyzed in Jolly (1965)

† `http://www.wildlifetrusts.org`

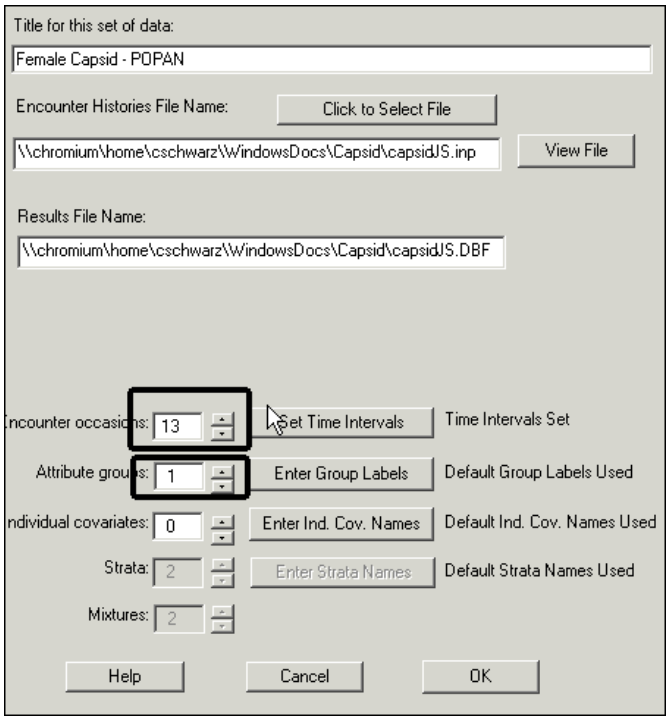
‡ We suspect that it was impossible to attach individually numbered tags to these small insects and some sort of batch marking scheme (e.g., color dots) was used. This batch marking scheme would enable the history of each insect to be followed, but individuals with the same history cannot be separately identified

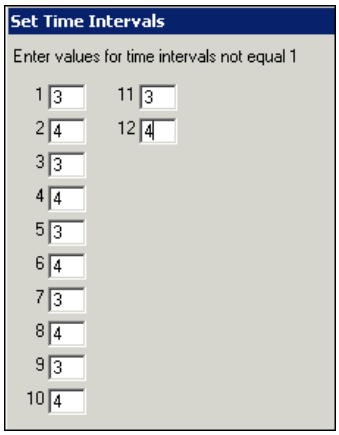
The summary statistics are:

Occasion	t_i	n_i	m_i	u_i	R_i	r_i	z_i
1	0	54	0	54	54	24	0
2	3	144	10	134	143	83	14
3	7	166	39	127	166	71	58
4	10	203	56	147	202	71	73
5	14	186	54	132	185	76	90
6	17	197	66	131	196	92	100
7	21	231	97	134	230	102	95
8	24	164	75	89	164	95	122
9	28	161	101	60	160	69	116
10	31	122	80	42	122	55	105
11	35	118	74	44	117	44	86
12	38	118	70	48	118	35	60
13	42	142	95	47	142	0	0

There are a few losses on capture at some of the sampling occasions where $n_i \neq R_i$.

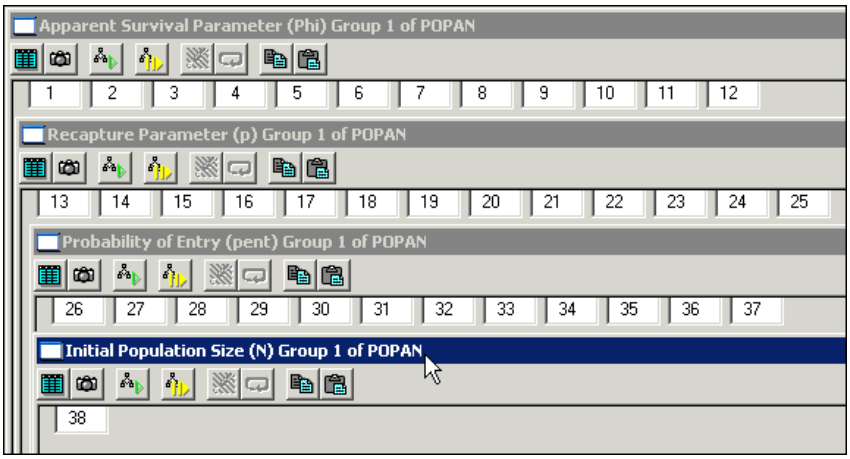
The data are input into **MARK** in the usual fashion. The time intervals are set to three and four day intervals in the usual fashion:



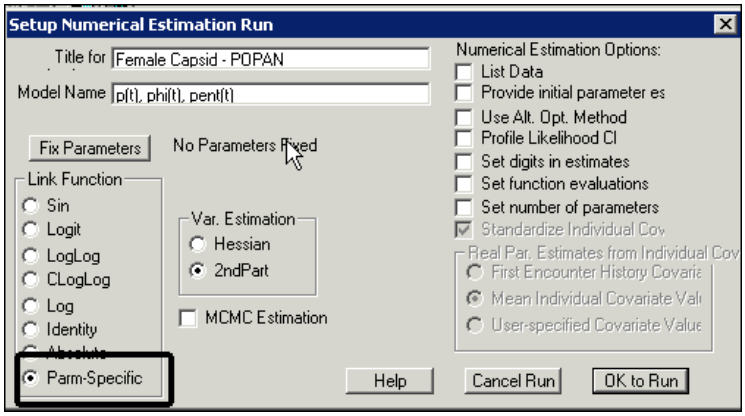


12.5.1. POPAN formulation

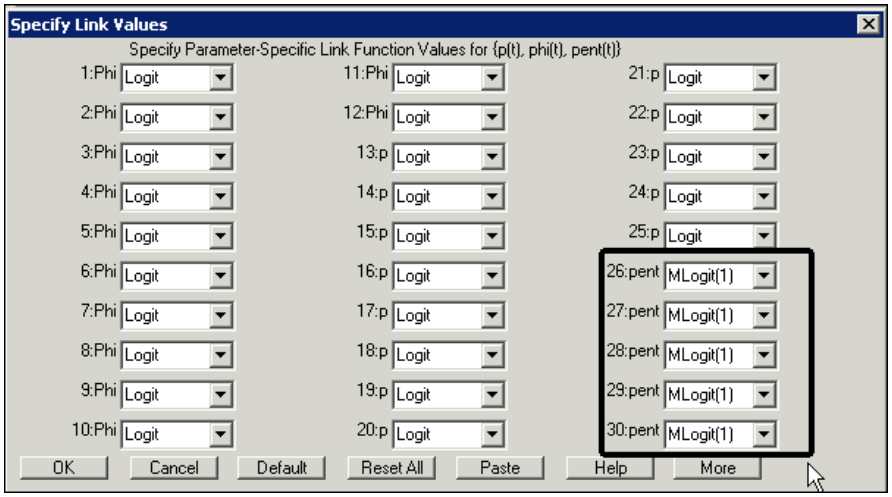
We begin by fitting the fully time-dependent model $\{p_t, \varphi_t, b_t\}$ in the usual fashion using PIMs:

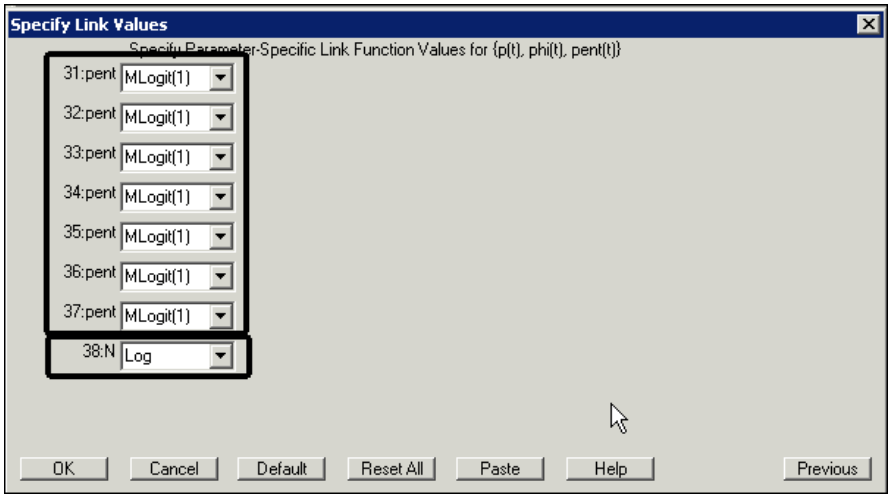


The model is run, again selecting the '**`parameter-specific link`**' functions:

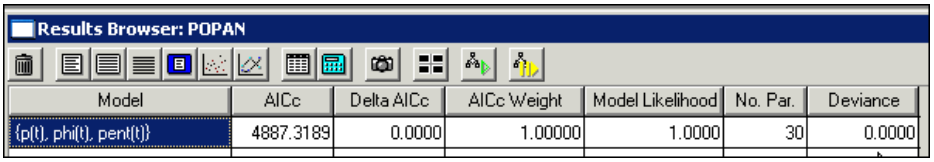


and specifying the `Mlogit(1)` link-function for the *PENT* parameters, and the *log* link-function for the super-population size (N).





The resulting model output is:

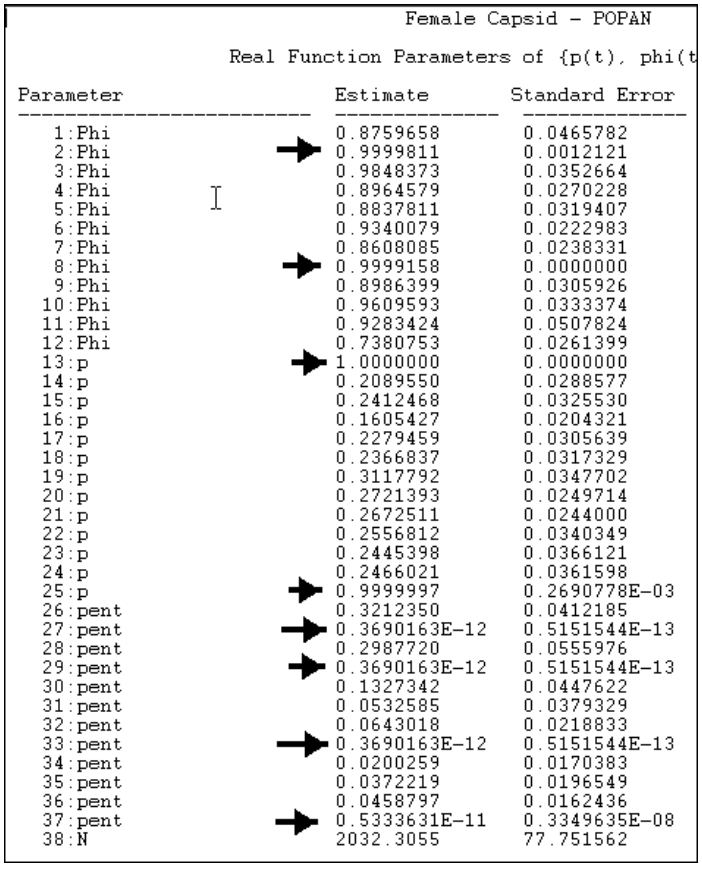


Model	AICc	Delta AICc	AICc Weight	Model Likelihood	No. Par.	Deviance
{p(t), phi(t), pent(t)}	4887.3189	0.0000	1.00000	1.0000	30	0.0000

There are a total of 36 parameters (13 capture probabilities; 12 survival probabilities; 13 entry probabilities [*], and 1 super-population size; less 2 confounded parameters at the start and end of the experiment

[*] Don't forget that **MARK** will show only the later 12 *PENT*s in the PIM and output because it never allows you to do anything with the b_0 parameter.

and less 1 restriction that the *PENT*s must sum to 1). The results browser only shows 30 parameters because some of the estimated *PENT*s, p's and φ's are estimated to be 0 or 1.*



Female Capsid - POPAN

Real Function Parameters of {p(t), phi(t

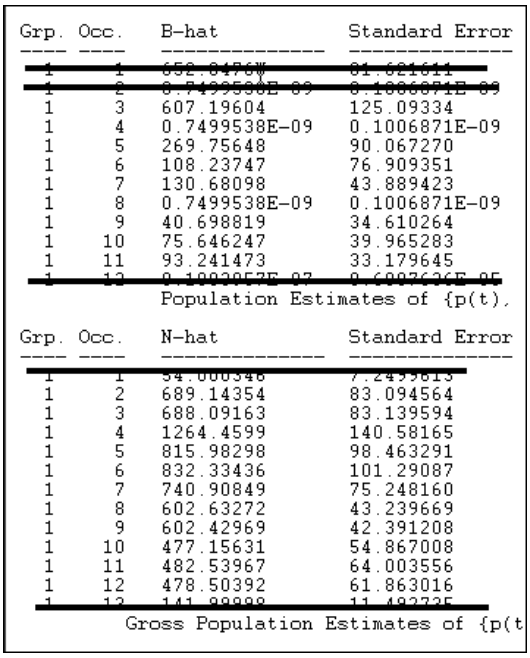
Parameter	Estimate	Standard Error
1:Phi	0.8759658	0.0465782
2:Phi	0.9999811	0.0012121
3:Phi	0.9848373	0.0352664
4:Phi	0.8964579	0.0270228
5:Phi	0.8837811	0.0319407
6:Phi	0.9340079	0.0222983
7:Phi	0.8608085	0.0238331
8:Phi	0.9999158	0.0000000
9:Phi	0.8986399	0.0305926
10:Phi	0.9609593	0.0333374
11:Phi	0.9283424	0.0507824
12:Phi	0.7380753	0.0261399
13:p	1.0000000	0.0000000
14:p	0.2089550	0.0288577
15:p	0.2412468	0.0325530
16:p	0.1605427	0.0204321
17:p	0.2279459	0.0305639
18:p	0.2366837	0.0317329
19:p	0.3117792	0.0347702
20:p	0.2721393	0.0249714
21:p	0.2672511	0.0244000
22:p	0.2556812	0.0340349
23:p	0.2445398	0.0366121
24:p	0.2466021	0.0361598
25:p	0.9999997	0.2690778E-03
26:pent	0.3212350	0.0412185
27:pent	0.3690163E-12	0.5151544E-13
28:pent	0.2987720	0.0555976
29:pent	0.3690163E-12	0.5151544E-13
30:pent	0.1327342	0.0447622
31:pent	0.0532585	0.0379329
32:pent	0.0643018	0.0218833
33:pent	0.3690163E-12	0.5151544E-13
34:pent	0.0200259	0.0170383
35:pent	0.0372219	0.0196549
36:pent	0.0458797	0.0162436
37:pent	0.5333631E-11	0.3349635E-08
38:N	2032.3055	77.751562

The number of parameters in the results browser should be reset to 36 in the usual fashion.

Because of the differing time intervals, the estimated survival rates (φ's) are the survival probabilities *per day*. They also differ from the estimates found in Jolly (1965) because they are also constrained to lie between 0 and 1. For example, Jolly (1965) estimated that the survival probability between the second and third sampling occasion was 1.015.

The estimated population sizes (shown at the top of the next page) are found in the *derived parameters* (accessed using '**Output** | **Specific Model Output** | **Parameter Estimates** | **Derived Estimates**'):

* This may be clearer if the β estimates table is also examined.



Grp.	Occ.	B-hat	Standard Error
1	1	652.8476W	81.621611
1	2	0.7499538E-09	0.1006871E-09
1	3	607.19604	125.09334
1	4	0.7499538E-09	0.1006871E-09
1	5	269.75648	90.067270
1	6	108.23747	76.909351
1	7	130.68098	43.889423
1	8	0.7499538E-09	0.1006871E-09
1	9	40.698819	34.610264
1	10	75.646247	39.965283
1	11	93.241473	33.179645
1	12	0.1003057E-07	0.6007636E-05

Population Estimates of {p(t),

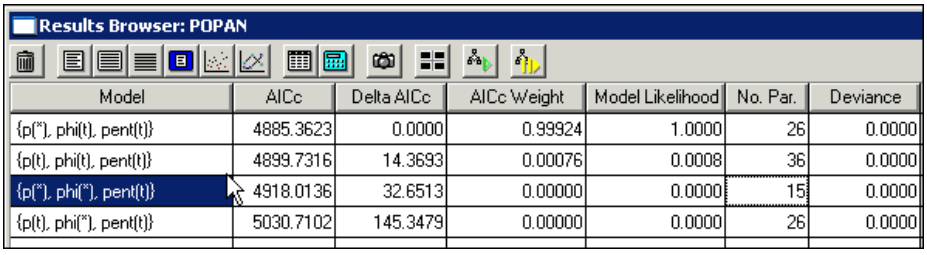
Grp.	Occ.	N-hat	Standard Error
1	1	54.000346	7.2499613
1	2	689.14354	83.094564
1	3	688.09163	83.139594
1	4	1264.4599	140.58165
1	5	815.98298	98.463291
1	6	832.33436	101.29087
1	7	740.90849	75.248160
1	8	602.63272	43.239669
1	9	602.42969	42.391208
1	10	477.15631	54.867008
1	11	482.53967	64.003556
1	12	478.50392	61.863016
1	13	141.99998	11.492735

Gross Population Estimates of {p(t

Because of confounding and non-identifiability at the start and the end of the experiment in the fully time-dependent model, some estimates cannot be used.

Model with constant p and/or constant φ *per day* may also be tenable and are fit in the usual way using the PIMs. Models with constant b's don't really have any sensible biological interpretation and should not be fit. A model for the emergence curve may be more sensible and this could result in predictions about the number of new entrants over time that has an early peak and tends to tail off over time. The number of parameters estimated by **MARK** should be checked and adjusted.

The final results table is:



Results Browser: POPAN

Model	AICc	Delta AICc	AICc Weight	Model Likelihood	No. Par.	Deviance
{p(*), phi(t), pent(t)}	4885.3623	0.0000	0.99924	1.0000	26	0.0000
{p(t), phi(t), pent(t)}	4899.7316	14.3693	0.00076	0.0008	36	0.0000
{p(*), phi(*), pent(t)}	4918.0136	32.6513	0.00000	0.0000	15	0.0000
{p(t), phi(*), pent(t)}	5030.7102	145.3479	0.00000	0.0000	26	0.0000

It appears that virtually all support lies on the model $\{p_\bullet, \varphi_t, b_t\}$. Because the capture probabilities are constant over time, there is no longer any problem with confounding or non-identifiability at the start or end of the experiment.

The final estimates of the basic parameters and the derived parameters are:

Real Function Parameters of {p(*), phi(t

Parameter	Estimate	Standard Error
1:Phi	0.8659472	0.0446034
2:Phi	0.9999992	0.1192227E-03
3:Phi	0.9392834	0.0254753
4:Phi	0.8934778	0.0202990
5:Phi	0.8956639	0.0249028
6:Phi	0.9594416	0.0187148
7:Phi	0.8503712	0.0180554
8:Phi	0.9997195	0.0000000
9:Phi	0.8953121	0.0186997
10:Phi	0.9556066	0.0232709
11:Phi	0.9533796	0.0303663
12:Phi	0.9999994	0.1128867E-03
13:p	0.2517871	0.0103360
14:pent	0.2425867	0.0208929
15:pent	0.3197505E-11	0.0000000
16:pent	0.1531475	0.0266547
17:pent	0.1220156	0.0280397
18:pent	0.1183510	0.0281414
19:pent	0.1003467	0.0292821
20:pent	0.0465839	0.0195714
21:pent	0.1570017E-06	0.6592920E-04
22:pent	0.0153315	0.0149972
23:pent	0.0350747	0.0161199
24:pent	0.0383922	0.0169404
25:pent	0.0217577	0.0167637
26:N	2014.6912	60.532767

Grp.	Occ.	B-hat	Standard Error
1	1	488.73722	43.307200
1	2	0.6441985E-08	0.0000000
1	3	308.54501	55.178680
1	4	245.82374	56.941222
1	5	238.44074	56.685513
1	6	202.16764	59.840812
1	7	93.852150	39.113762
1	8	0.3163099E-03	0.1328281
1	9	30.888278	30.180443
1	10	70.664776	32.448409
1	11	77.348430	34.059597
1	12	43.835065	34.099976

Population Estimates of {p(*),

Grp.	Occ.	N-hat	Standard Error
1	1	214.38783	30.097406
1	2	627.94845	40.703996
1	3	626.94640	40.704221
1	4	828.08666	57.929862
1	5	772.91542	54.495706
1	6	793.07278	54.416236
1	7	873.34802	57.400075
1	8	630.28498	29.287469
1	9	629.57830	13.944050
1	10	481.99858	38.455100
1	11	472.60577	39.367002
1	12	486.02212	40.529796
1	13	529.85603	40.558755

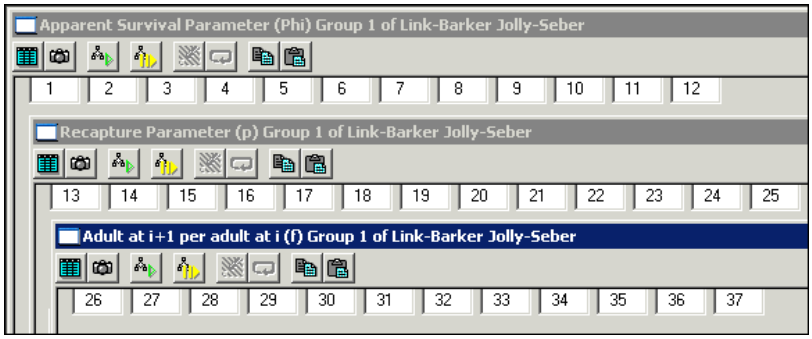
The estimated super-population size is interpreted as the total number of capsids ever present in the experiment and does not represent the number present at any particular point in time. The population seems to peak at about sampling occasion 4, and then gradually tapers off because of deaths/emigration and fewer new insects entering the population of interest.

12.5.2. Link-Barker and Pradel-recruitment formulations

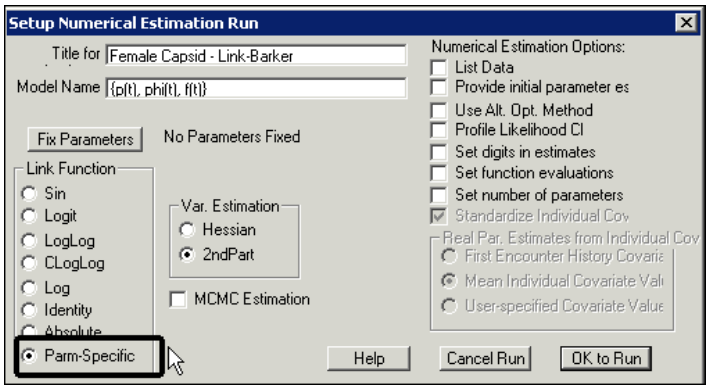
Both the Link-Barker and Pradel-recruitment parameterize new entrants to the population using the f_i parameter representing the numbers of new recruits in the interval per member of the population alive at time i. Because this dataset includes losses-on-capture, the Pradel-recruitment model cannot be used, and the Link-Barker formulation will be used.

The data are entered as shown above – don't forget to set the alternating 3- and 4-day time intervals.

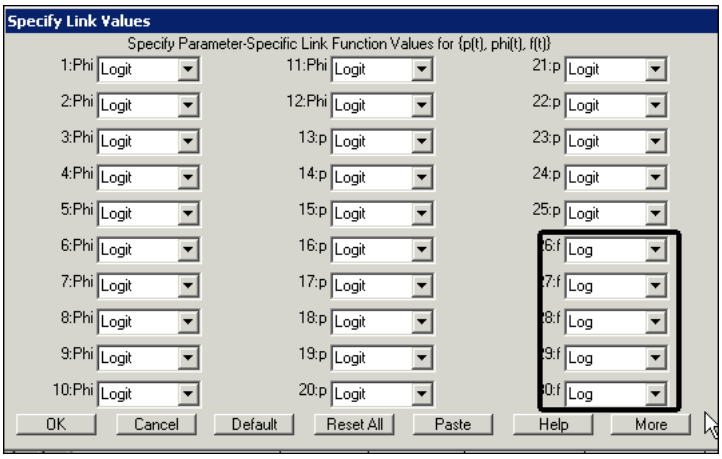
The fully time-dependent model $\{p_t, \varphi_t, f_t\}$ is fit using PIMs in the usual fashion:

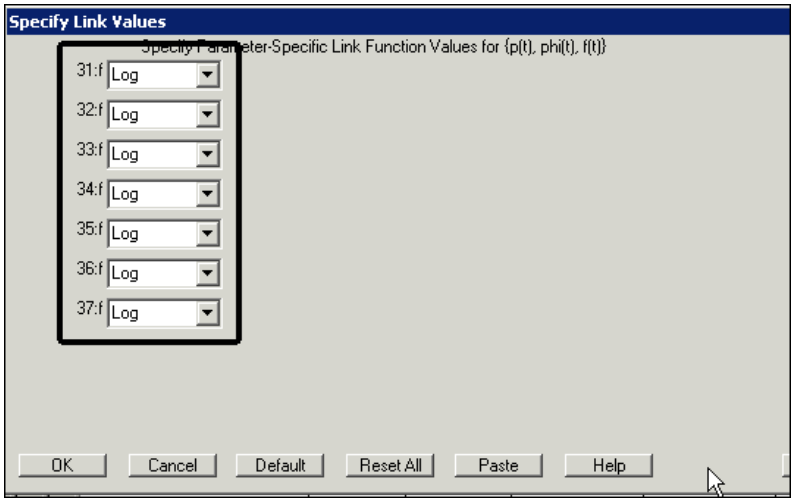


This model is run in the usual fashion.

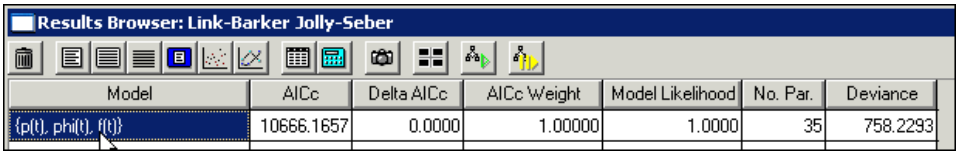


The recruitment parameter should have a *log* link-function specified as this parameter can exceed the value of 1.





The results table is:



There are 35 parameters (13 capture probabilities; 12 survival probabilities; 12 recruitment probabilities; less 1 non-identifiable parameter at the end of the sampling chain where $\varphi_{12}p_{13}$ can only be estimated; and less 1 non-identifiable parameter at the start of the sampling chain.) If the number of parameters in the results table differs, it should be changed in the usual fashion.

The estimates from this model are shown at the top of the next page. The estimated survival probabilities and recruitment parameters are on a *per day* basis. The estimates of p and φ are comparable to the *POPAN* estimates except for some minor differences at the start of the sampling chain. These are artifacts of the different confounding at the start of the sampling chain between the two formulations. The

estimated recruitment parameters indicate the number of new recruits per member of the population. Estimates of the actual population size are not available.

Real Function Parameters of {p(t),

Parameter	Estimate	Standard Err
1:Phi	1.0000000	0.0000000
2:Phi	0.9830163	0.0259774
3:Phi	0.9562097	0.0394007
4:Phi	0.9035631	0.0277166
5:Phi	0.8838988	0.0319599
6:Phi	0.9339469	0.0223125
7:Phi	0.8616390	0.0255083
8:Phi	0.9976477	0.0229306
9:Phi	0.9003750	0.0355637
10:Phi	0.9609894	0.0333523
11:Phi	0.9235196	0.0512547
12:Phi	0.7373640	0.0201540
13:p	0.1033227	0.0000000
14:p	0.1821002	0.0056901
15:p	0.2241029	0.0371545
16:p	0.2123710	0.0337424
17:p	0.1977410	0.0306743
18:p	0.2365201	0.0317549
19:p	0.3116825	0.0347872
20:p	0.2714373	0.0259713
21:p	0.2689865	0.0305414
22:p	0.2556698	0.0340466
23:p	0.2444797	0.0366274
24:p	0.2570973	0.0431501
25:p	1.0000000	0.0000000
26:f	0.0006197	0.0000000
27:f	0.2533830	0.0000000
28:f	0.7462776	0.1289553
29:f	0.7508969	0.0936962
30:f	0.5802783	0.1278921
31:f	0.6000752	0.1197371
32:f	0.5603289	0.0761922
33:f	0.0648895	1.6747656
34:f	0.4088284	0.1210313
35:f	0.6304078	0.0917065
36:f	0.5477003	0.1062975
37:f	0.3357953	0.1732492

Simpler models for p and φ can be fit in the usual fashion. It may be actually biologically sensible to fit a model with constant f over time as this is the recruitment per existing member. Even if the population is declining over time, perhaps the recruitment is constant over time. The results table for these models is:

Results Browser: Link-Barker Jolly-Seber

Model	AICc	Delta AICc	AICc Weight	Model Likelihood	No. Par.	Deviance
{p(*), phi(t), f(t)}	10657.4228	0.0000	0.98752	1.0000	25	770.1116
{p(t), phi(t), f(t)}	10666.1657	8.7429	0.01248	0.0126	35	758.2293
{p(*), phi(*), f(t)}	10683.1907	25.7679	0.00000	0.0000	14	818.3268
{p(t), phi(*), f(t)}	10783.0539	125.6311	0.00000	0.0000	25	895.7421
{p(*), phi(t), f(*)}	10806.0321	148.6093	0.00000	0.0000	15	939.1373

Virtually all support again lies with the model with constant catchability $\{p_\bullet, \varphi_t, f_t\}$. Because capture probabilities were constant over time, all parameters are now estimable. The final estimates are:

Female Capsid - Link-Barker

Real Function Parameters of {p(*), ph

Parameter	Estimate	Standard Error
1:Phi	1.0000000	0.1128761E-04
2:Phi	0.9727813	0.0207076
3:Phi	0.9401501	0.0285802
4:Phi	0.8934947	0.0203122
5:Phi	0.8957113	0.0249229
6:Phi	0.9594949	0.0187335
7:Phi	0.8502892	0.0208828
8:Phi	0.9999810	0.0000000
9:Phi	0.8952598	0.0238415
10:Phi	0.9556793	0.0232962
11:Phi	0.9535965	0.0304135
12:Phi	0.9999999	0.2860069E-04
13:p	0.2512138	0.0105578
14:f	1.2010867	0.0924535
15:f	0.7083365	0.0831203
16:f	0.7377478	0.0654526
17:f	0.7378570	0.0477018
18:f	0.6751899	0.0600519
19:f	0.7101521	0.0579573
20:f	0.4743710	0.0711472
21:f	0.0486908	1.7038972
22:f	0.3648598	0.1219773
23:f	0.6180099	0.0748742
24:f	0.5460166	0.0851689
25:f	0.5476497	0.1116569

The estimates of p and φ are comparable to those from the *POPAN*formulation except at the start of the sampling chain. This may be an artifact of convergence to a local minimum by **MARK**. The estimates of recruitment can be matched to that from *POPAN*. For example,

$$\hat{f}_{12}^{LB} = 0.54765, \hat{B}_{12}^{POPAN} = 43.83, \text{and } \hat{N}_{12}^{POPAN} = 486.02.$$

Now

$$\frac{\hat{B}_{12}^{POPAN}}{\hat{N}_{12}^{POPAN}} = \frac{43.83}{486.02} = 0.090 = (\hat{f}_{12}^{LB})^4 = 0.548^4 = 0.090$$

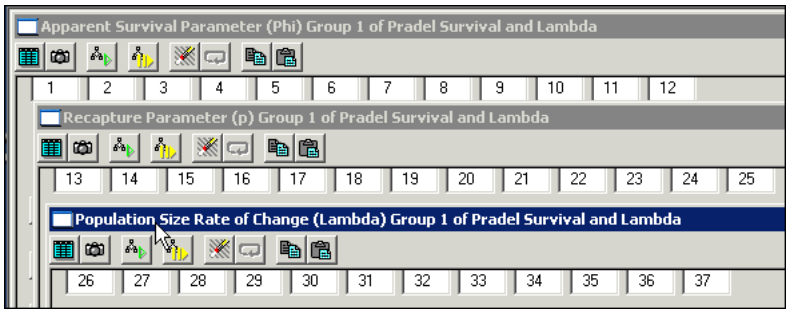
12.5.3. Burnham Jolly-Seber and Pradel-λ formulations

The Burnham-Jolly-Seber and the Pradel-λ formulations parameterize new recruits to the population indirectly by estimating population growth (λ) representing the population size at time $i + 1$ relative to the population size at time i. The growth process is the net effect of both survival and recruitment.

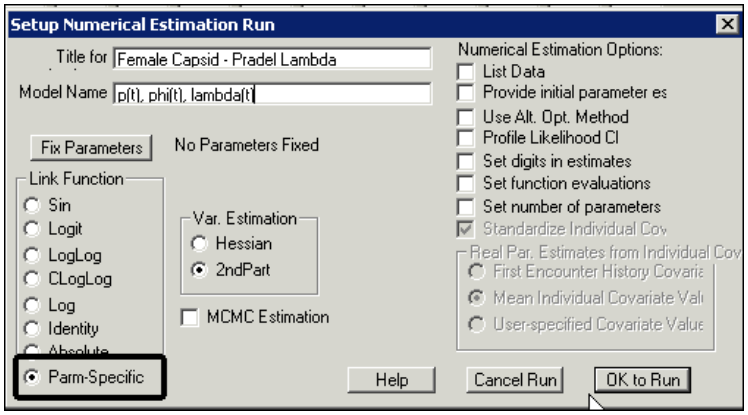
The data are entered in the usual fashion - don't forget to set the alternating 3- and 4-day intervals.

The Burnham-Jolly-Seber formulation again has difficulty in convergence for this example and so is not run against this data.

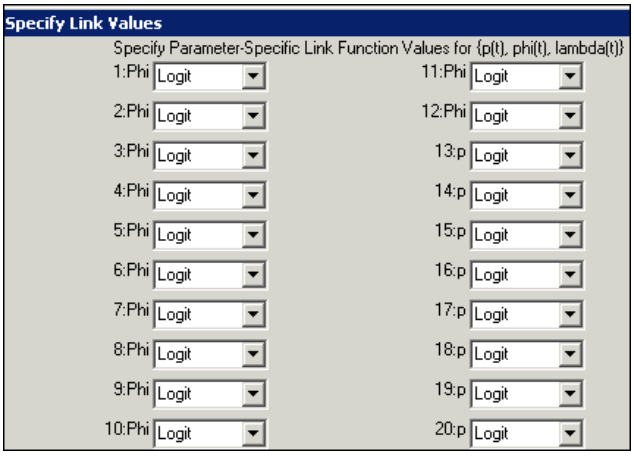
The fully time-dependent Pradel-λ model $\{p_t, \varphi_t, \lambda_t\}$ is fit using PIMs in the usual fashion:

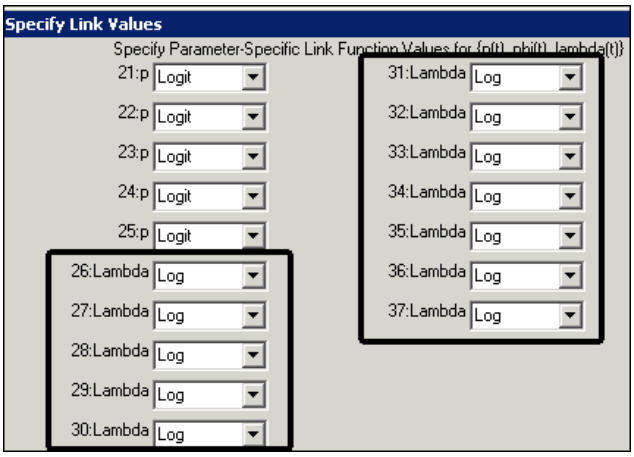


This model is run in the usual fashion.

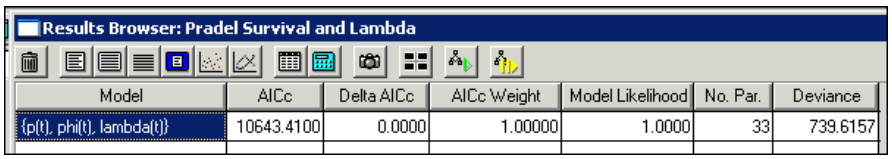


The population growth parameter should have a *log* link-function specified as this parameter can exceed the value of 1.





The results table is:



Results Browser: Pradel Survival and Lambda

Model	AICc	Delta AICc	AICc Weight	Model Likelihood	No. Par.	Deviance
{p(t), phi(t), lambda(t)}	10643.4100	0.0000	1.00000	1.0000	33	739.6157

There are 35 parameters (13 capture probabilities; 12 survival probabilities; 12 growth rates; less 1 non-identifiable parameter at the end of the sampling chain where $\varphi_{12}p_{13}$ can only be estimated; and less 1 non-identifiable parameter at the start of the sampling chain.) If the number of parameters in the results table differs, it should be changed in the usual fashion.

The estimates from this model are:

Parameter	Estimate	Standard Error
1:Phi	0.8622702	0.0475434
2:Phi	0.9999994	0.1468950E-03
3:Phi	0.9637049	0.0349087
4:Phi	0.9035631	0.0277154
5:Phi	0.8838963	0.0319597
6:Phi	0.9339488	0.0223132
7:Phi	0.8630991	0.0259921
8:Phi	0.9968961	0.0242281
9:Phi	0.9001734	0.0360752
10:Phi	0.9609885	0.0333552
11:Phi	0.9235251	0.0512570
~~12:Phi~~	~~0.7937336~~	~~4.5424526~~
~~13:p~~	~~0.3537664~~	~~11.156657~~
14:p	0.2888521	0.0857385
15:p	0.2326688	0.0332320
16:p	0.2123706	0.0337403
17:p	0.1977443	0.0306722
18:p	0.2365221	0.0317525
19:p	0.3116810	0.0347862
20:p	0.2625937	0.0311958
21:p	0.2729863	0.0324555
22:p	0.2556655	0.0340469
23:p	0.2444797	0.0366279
24:p	0.2570829	0.0431528
~~25:p~~	~~0.7472040~~	~~17.106092~~
~~26:Lambda~~	~~1.4036600~~	~~15.597255~~
27:Lambda	1.0952552	0.0833055
28:Lambda	1.1023949	0.0713553
29:Lambda	0.9965351	0.0509409
30:Lambda	0.9608886	0.0594949
31:Lambda	0.9716717	0.0377260
32:Lambda	0.9450045	0.0445205
33:Lambda	0.9857956	0.0341077
34:Lambda	0.9323707	0.0460851
35:Lambda	1.0027807	0.0421602
36:Lambda	0.9839781	0.0645604
~~37:Lambda~~	~~0.9931150~~	~~4.5094713~~

The estimated survival probabilities and growth rates are on a *per day* basis. The estimates of p and φ are comparable to the *POPAN* estimates except for some minor differences at the start of the sampling chain. These are artifacts of the different confounding at the start of the sampling chain between the two formulations. The estimated growth parameters indicate ratio of the estimated population size at successive sampling intervals on a per unit time basis. Estimates of the actual population size are not available directly, but as illustrated in previous examples can be derived if needed.

Simpler models for p and φ can be fit in the usual fashion. It may be actually biologically sensible to fit a model with constant λ over time if the population is roughly constant over time. Because the growth rate includes both survival and recruitment, models where growth is constant over time, but survival is not, are not usually fit as it is difficult to believe that changes in recruitment will exactly balance changes in survival to keep the population at a constant level. The results table for these simpler models is:

Results Browser: Pradel Survival and Lambda

Model	AICc	Delta AICc	AICc Weight	Model Likelihood	No. Par.	Deviance
{p(*), phi(t), lambda(t)}	10637.3658	0.0000	0.99386	1.0000	25	750.0546
{p(t), phi(t), lambda(t)}	10647.5518	10.1860	0.00610	0.0061	35	739.6157
{p(t), phi(*), lambda(t)}	10657.6359	20.2701	0.00004	0.0000	25	770.3246
{p(*), phi(*), lambda(t)}	10674.0657	36.6999	0.00000	0.0000	14	809.2009

Virtually all support again lies with the model with constant catchability $\{p_\bullet, \varphi_t, \lambda_t\}$. Because capture rates were constant over time, all parameters are now estimable. The final estimates are:

Parameter	Estimate	Standard Error
1:Phi	0.8651740	0.0445676
2:Phi	0.9999729	0.0000000
3:Phi	0.9384308	0.0253764
4:Phi	0.8932861	0.0202472
5:Phi	0.8954453	0.0248346
6:Phi	0.9591061	0.0186605
7:Phi	0.8500244	0.0208551
8:Phi	0.9999927	0.7759302E-03
9:Phi	0.8947431	0.0238164
10:Phi	0.9554366	0.0231951
11:Phi	0.9525612	0.0302881
12:Phi	0.9999997	0.7881691E-04
13:p	0.2533691	0.0105643
14:Lambda	1.3830550	0.0698444
15:Lambda	1.0451863	0.0241858
16:Lambda	1.0717819	0.0299564
17:Lambda	0.9835025	0.0204745
18:Lambda	1.0093050	0.0276135
19:Lambda	1.0248589	0.0198782
20:Lambda	0.9043622	0.0240825
21:Lambda	0.9899406	0.0162655
22:Lambda	0.9204955	0.0258054
23:Lambda	0.9949499	0.0237861
24:Lambda	1.0095378	0.0321254
25:Lambda	1.0219978	0.0171985

The estimates of p and φ are comparable to those from the *POPAN* formulation. There are only a few sampling occasions where the estimates of population growth are inconsistent with estimates of survival, but the differences are minor.

The estimates of population can be matched to that from *POPAN* . For example, $\hat{\lambda}_{12}^{PL} = 1.0219978$, $\hat{N}_{12}^{POPAN} = 486.02$, and $\hat{N}_{13}^{POPAN} = 529.86$.

Now

$$\frac{\hat{N}_{13}^{POPAN}}{\hat{N}_{12}^{POPAN}} = \frac{529.86}{486.02} = 1.09 = (\hat{\lambda}_{12}^{PL})^4 = 1.0219978^4 = 1.09$$

12.6. Final words

While many researchers think of population numbers and recruitment in terms of actual animals entering populations, the JS model can be extended in a number of ways:

- Manske and Schwarz (2000) used a Jolly-Seber model to estimate stream residence times of salmon. This extended the work of Schaub *et al*. (2001) who used mark-recapture methods to estimate stop-over times of migrating birds. In both methods, the population is transient with new animals arriving and departing on regular basis and the average time at the sampling location is of interest. The key difference between the two approaches is that the methods of Schaub *et al*. (2001) assume that the day the animal is marked is the first day of residence while Manske and Schwarz (2000) did not make this assumption.
- Schwarz and Arnason (2000) and Manske *et al*. (2002) showed how to use the *POPAN* parametrization to estimate age-specific breeding proportions, i.e., what fraction of animals enter the breeding population at each age.

While current implementations of the JS model allow for multiple groups (e.g., males and females), animals are not allowed to change groups during the experiment. The Cormack-Jolly-Seber (CJS) model has been extended to a multi-state version where animals are allowed to change states during the experiment (e.g., geographical movement) and this is discussed in detail in Chapter 10. Recently, Dupuis and Schwarz (2007) have extended the Jolly-Seber model to allow multiple states. In their example, they modeled a fish population that spawned at various locations around a lake and moved among spawning location during the multiple years of the study. Estimates of abundance at each spawning location and recruitment to spawning locations were obtained.

This stratified Jolly-Seber model can also be used to model stratified closed populations and the Jolly-Seber age-structured model.

The examples in this chapter did not use covariates. System-wide covariates that affect all animals at a particular time point (e.g., temperature) are easily implemented using design matrices.

One area that requires further work is the use of individual covariates. Individual covariates take two forms - those that vary among individuals, but are fixed for the individual for the study, and individual time-varying covariates. There are two major difficulties. First, even if the covariates are fixed for each animal for the entire study, the value of the covariate is unknown if the animal is not seen. Second, if the individual covariates can change values during the experiment, the value of the covariate is also unknown when an animal is not recaptured after being captured for the first time.

McDonald and Amstrup (2001) used a Horvitz-Thompson type estimator to incorporate individual fixed covariates and were able to estimate population sizes. However, this approach does not have a likelihood basis and so multi-group methods where restrictions on parameters are placed across groups is not easily implemented. Bonner and Schwarz (2006) recently developed methods for the CJS model for individual time-varying covariates, but this has not been extended to JS models. Stay tuned for developments over the next few years.

References

Arnason, A. N. and Schwarz, C. J. (1995) POPAN-4. Enhancements to a system for the analysis of mark-recapture data from open populations. *Journal of Applied Statistics* **22**, 785-800.

Arnason, A. N., Schwarz, C. J. and Boyer, G. (1998) POPAN-5: A data maintenance and analysis system for mark recapture data. Technical Report, Department of Computer Science, University of Manitoba viii+318 p.

Arnason, A. N. and Schwarz, C. J. (1999) Using POPAN-5 to analyze banding data. *Bird Study* **46** (suppl), s157-s168.

Arnason, A. N. and Schwarz, C. J. (2002) POPAN-6: exploring convergence and estimate properties with SIMULATE. *Journal of Applied Statistics* **29**, 649-668.

Bonner, S. J. and Schwarz, C. J. (2006) An Extension of the Cormack-Jolly-Seber Model for Continuous Covariates with Application to *Microtus pennsylvanicus*. *Biometrics* **62**, 142-149.

Burnham, K. P. (1991) On a unified theory for release-resampling of animal populations. In *Proceedings of 1990 Taipei Symposium in Statistics*, M. T. Chao and P. E. Cheng (eds), 11-36. Institute of Statistical Science, Academia Sinica: Taipei, Taiwan.

Cormack, R. M. (1964) Estimates of survival from the sighting of marked animals. *Biometrika* **51**, 429-438.

Cowen, L. L. and Schwarz, C. J. (2006) The Jolly-Seber model with tag-loss. *Biometrics* **62**, 699-705.

Crosbie, S. F. and Manly, B. F. (1985) Parsimonious modeling of capture-mark-recapture studies. *Biometrics* **41**, 385-398.

Dupuis, J. A. and Schwarz, C. J. (2007) A Bayesian approach to the multistate Jolly-Seber capture-recapture model. *Biometrics* **63**, 1015-1022.

Jolly, G. M. (1963) Estimates of population parameters from multiple recapture data with both death and dilution - deterministic model. *Biometrika* **50**, 113-128.

Jolly, G. M. (1965) Explicit estimates from capture-recapture data with both death and immigration - Stochastic model. *Biometrika* **52**, 225-247.

Lebreton, J.-D., Burnham, K. P., Clobert, J. and Anderson, D. R. (1992) Modeling survival and testing biological hypotheses using marked animals. A unified approach with case studies. *Ecological Monographs* **62**, 67-118.

Link, W. A. and Barker, R. J. (2005) Modeling association among demographic parameters in analysis of open population capture-recapture data. *Biometrics* **61**, 46-54.

Manske, M. and Schwarz, C. J. (2000) Estimates of stream residence time and escapement based on capture-recapture data. *Canadian Journal of Fisheries and Aquatic Sciences* **57**, 241-246.

Manske, M., Stobo W. T., and Schwarz, C. J. (2002) Estimation of age-specific probabilities of first return and annual survival probabilities for the male gray seal (*Halichoerus grypus*) on Sable Island from capture-recapture data. *Marine Mammal Science* **18**, 145-155.

McDonald, T. L. and Amstrup, S. C. (2001) Estimation of Population Size Using Open Capture-Recapture Models. *Journal of Agricultural, Biological and Environmental Statistics* **6**, 206-220.

Muir, R. C. (1958) On the application of the capture-recapture method to an orchard population of *Blepharidopterus angulatus* (Fall.) (*Hemiptera-Heteroptera, Miridae*). *Report of the East Malling Research Station* **1959**, 140-47.

Pledger, S. and Efford, M. (1998) Correction of bias due to heterogeneous capture probability in capture-recapture studies of open populations. *Biometrics* **54**, 888-898.

Pradel, R. (1996) Utilization of capture-mark-recapture for the study of recruitment and population growth rate. *Biometrics* **52**, 703-709

Rotella, J. J. and Hines, J. E. (2005) Effects of tag loss on direct estimates of population growth rate. *Ecology* **86**, 821-827.

Sanathanan, L. P. (1972) Estimating the size of a multinomial population. *Annals of Mathematical Statistics* **43**, 142-152.

Sanathanan, L. P. (1977) Estimating the size of a truncated sample. *Journal of the American Statistical Association* **72**, 669-672.

Schaub M., Pradel R., Jenni, L. and Lebreton J.-D. (2001) Migrating birds stop over longer than usually thought: an improved capture-recapture analysis. *Ecology* **82**, 852-859.

Schwarz, C. J. (2001) The Jolly-Seber Model: More Than Just Abundance. *Journal of Agricultural, Biological and Environmental Statistics* **6**, 195-205.

Schwarz, C. J. and Arnason, A. N. (1996) A general methodology for the analysis of open-model capture recapture experiments. *Biometrics* **52**, 860-873.

Schwarz, C. J. and Arnason, A. N. (2000) The estimation of age-specific breeding probabilities from capture-recapture data. *Biometrics* **56**, 59-64.

Schwarz, C. J., Bailey, R.,E., Irvine, J. R. and Dalziel, F. C. (1993) Estimating salmon spawning escapement using capture-recapture methods. *Canadian Journal of Fisheries and Aquatic Sciences* **50**, 1181-1191.

Seber, G. A. F. (1965) A note on the multiple recapture census. *Biometrika* **52**, 249-259.

Stanley, T. R. and Burnham, K. P., (1999) A goodness-of-fit test for capture-recapture model Mt closure. *Biometrics* **55**, 366-375.

Sykes, S. D. and Botsford, L. W. (1986) Chinook salmon, *Oncorhynchus tshawytscha*, spawning escapement based upon multiple mark-recapture of carcasses. *Fisheries Bulletin* **84**, 261-270.

Williams, B K., J. D. Nichols, and M. J. Conroy. (2002) *Analysis and management of animal populations: modeling, estimation, and decision making*. Academic Press, New York.

CHAPTER 13

Pradel models: recruitment, survival and population growth rate

So far, we have concentrated on estimation related to the general question 'what is the probability of leaving the population?'. Clearly, death marks permanent departure from the population. Absence from the population can be permanent (like death), or temporary (a subject we'll discuss more fully in a later chapter on something known as the 'robust design'). However, if we're interested in modeling the dynamics of a population, then we're likely to be as interested in the probability of entry into the population as we are the probability of exit from the population. So, where to begin. We'll start with the fundamental model of population dynamics. Usually, the assumption (based on even a casual glance at a typical textbook on the subject) is that population dynamics models are based entirely on high-level mathematics. However, while it isn't difficult to find examples of such models, the fundamental model is quite simple:

population dynamics has to do with the change in abundance over space and time (ΔN)

ΔN = 'additions' - 'subtractions'

That's it, really. The rest is just 'details' (of course, the details can get messy, and that is what often leads to the higher math referred to above). But the basic idea is simple: when the net number of additions is greater than the net number of subtractions, then clearly the population will grow ($\Delta N > 0$). When the reverse is true, the population will decline. So, population dynamics involves the study and estimation of the relative contributions to the 'additions' and 'subtractions' from the population.

13.1. Population growth: realized vs. projected

Usually, the net growth of a population is expressed as the ratio of successive population abundances: N_{t+1}/N_t. This ratio is usually referred to as λ, leading (frequently) to a whole bunch of confusion - λ as the ratio of successive population sizes, or λ as the projected growth rate of a population under specified model conditions? As you may recall, the *projected* growth of a structured population is given as the dominant eigenvalue from a non-negative projection matrix (the 'Leslie' matrix for models structured on age, and the 'Lefkovitch' matrix for situations where some other classification factor – often size or developmental state – is a better demographic category than age). The word 'projected' is key here – it is

© Cooch & White (2014)

the growth rate of the population that would eventually be expected if (and only if) the conditions under which the model are valid are time invariant (i.e., not stochastic). We differentiate between *projected* λ and *realized* λ. We let realized λ be (simply) the ratio of successive population sizes. Projected λ and realized λ will be equivalent if and only if the growth of the population has achieved stationary, ergodic conditions (the familiar stable-age or stable-age structure at equilibrium). Under such conditions, the population (and each age-class in the population) will be growing at rate λ, such that $\lambda = N_{t+1}/N_t$. So, we suggest qualifying the used of 'λ' with a prefix - either projected, or realized, and noting (if appropriate) when they are equivalent, and when they are different:

projected λ: the growth rate of the population expected under time invariance (where λ is commonly derived as the dominant eigenvalue from a projection matrix model)

realized λ: the observed growth rate of the population between successive samples (time steps): $\lambda_i = N_{i+1}/N_i$

While projected λ is often of considerable interest (especially for prospective management studies), in a retrospective study, where we're interested in assessing the pattern of variation in growth that has occurred, it is realized λ that is of interest (although there are a variety of analytical methods out there for retrospective analysis that somewhat blur this convenient distinction: the life table response experiment (LTRE) developed by Caswell is a hybrid of a retrospective technique using prospective perturbation analysis of deviation in the projected λ under a variety of experimental conditions). For our purposes though, we'll keep to the simple distinction we've just described – we want to explore changes in *realized* growth, λ.

13.2. Estimating realized λ

Since $\lambda = N_{t+1}/N_t$, then it seems reasonable that as a first step, we want to derive estimates of abundance for our population at successive time steps, and simply derive the ratio to yield our estimate of λ. Simple in concept, but annoyingly difficult in practice. Why? In large part, because the estimation of abundance in an open population is often very imprecise. Such estimates often suffer rather profoundly from violation of any of a number of assumptions, and are often not worth the effort (abundance estimation in open populations was covered in Chapter 12 – estimation in closed populations is covered in Chapter 14).

So we're stuck, right? Not exactly. The 'solution' comes in several steps. We start with the recognition that our basic purpose in characterizing the change in abundance between years rests on assessing the relative number of 'additions' and 'subtractions'. This basic idea was introduced in Chapter 12. With a bit of thought, you should realize that an individual which 'dies', or 'permanently emigrates', is clearly a 'subtraction' from the population. As such, we can reasonably state that the number of individuals in the population next year is going to be a function, at least in part, of the number of individuals in the population this year that survive and return to the population next year.

However, we also know that there may be 'additions' to the population, either in the form of permanent immigration, or births (*in situ* recruits). Let the number of individuals this year be N_t. Let φ_i be the probability of surviving and returning to the population ($\varphi_i = S_i F_i$, where S is the true survival probability , and F is the fidelity probability – see chapter 10). Let B_i be the number of new individuals that enter the population between (i) and (i+1) - in other words, B_i is the number of individuals in the population at (i+1) that were **not** there at (i). Thus, we can write:

$$N_{i+1} = N_i \varphi_i + B_i$$

Next, some simple algebra. First, recall we define λ as the ratio of successive population sizes. Thus, $\lambda_i = N_{i+1}/N_i$. Thus, substituting this into the previous expression, and doing a bit of re-arranging, we get:

$$\lambda_i = \frac{B_i}{N_i} + \varphi_i$$

Now, λ and φ are familiar (and explicitly defined above). What about B_i/N_i? This is the per capita rate of additions to the population (often referred to somewhat 'sloppily' as the recruitment rate, which has a very specific demographic meaning that is often ignored – for purposes of consistency with some of the literature, we'll ignore it too). It is the number of individuals entering the population between (i) and (i+1) (i.e., B_i) per individual already in the population at time (i) (i.e., N_i). Let's call this recruitment probability f_i. Thus, we write

$$\begin{aligned}\lambda_i &= \frac{B_i}{N_i} + \varphi_i \\ &= f_i + \varphi_i\end{aligned}$$

OK, so far so good. But perhaps right about now you're asking yourself 'well, how does this help?'. We can estimate φ_i fairly well (as discussed in the first several chapters of this guide), but what about this recruitment thing, f_i? After all, f_i is B_i/N_i, both of which we've just implied are difficult to estimate with any precision in an open population.

13.2.1. Reversing encounter histories: φ and γ

Ah, now for the **big** 'trick', which is so intuitively obvious once we describe it we should probably pause long enough for you to slap yourself in the forehead. Back in 1965, George Jolly, and later Ken Pollock in 1974, realized that the encounter histories carried a lot more information than we often realize. They basically noted that

> '*if estimating the transitions among encounter occasions going forward in time yields an estimate of the probability of remaining alive and in the population,* φ*, where* $(1 - \varphi)$ *is the probability of* leaving *the population, then if you simply reverse the encounter histories, the transition parameter being estimated is the probability of* entering *the population*'.

Why do we care (or, cute trick, but is it useful)? We care (and find utility) because that's precisely what we're after. Recruitment is the process of entering the population. So, if we had a parameter that allowed us to estimate the probability of entering a population (i.e., recruiting), then we're clearly on the right track.

In fact, this is precisely what Roger Pradel describes in his 1996 paper.* Re-discovering (and extending) the earlier work of Jolly and Pollock, Pradel explicitly noted the duality between analyzing the encounter history going forward, and going backward in time. He introduced a parameter γ_i (which he refered to as the *seniority* parameter), which he defined as 'the probability that if an individual is alive and in the population at time i that it was also alive and in the population at time i-1'.

* Pradel, R. (1996) Utilization of capture-mark-recapture for the study of recruitment and population growth rate. *Biometrics*, **52**, 703-709.

Let's pause to highlight the distinctions between φ_i (going forward in time) and γ_i (estimated from the reverse encounter history, going backward in time):

forward in time	φ_i	probability that if alive and in the population at time *i* (e.g., this year), the you will be alive and in the population at time *i*+1 (e.g., next year)
backward in time	γ_i	probability that if alive and in the population at time *i* (e.g., this year), that you *were* also alive and in the population at time *i*-1 (e.g., last year)

begin sidebar

Understanding 'backwards' encounter histories

Consider the following encounter history: '101110'. What this history is telling us, going forward in time, is that the individual was initially encountered and marked at occasion 1, not seen at occasion 2, but then seen on the next 3 occasions (3, 4, and 5), then not seen on occasion 6. The probability expression corresponding to this history would be:

$$\varphi_1(1-p_2)\varphi_2 p_3 \varphi_3 p_4 \varphi_4 p_5 (1-\varphi_5 p_6)$$

Now, if we reverse the encounter history ('101110' → '011101'), then we see that, conditional on being alive and in the population at occasion 5, that the individual was also in the population at occasion 4, and at occasion 3. Given that it was also there at occasion 1, but not encountered at occasion 2 (with probability $1-p_2$), then the probability expression (in terms of γ_i and p_i) corresponding to '011101' is

$$\gamma_5 p_4 \gamma_4 p_3 \gamma_3 (1-p_2) \gamma_2 p_1$$

end sidebar

13.2.2. Putting it together: deriving λ

Still with us? We hope so, because the parameter γ_i features prominently in what comes next. Remember, our interest in this parameter γ_i comes from it having something to do with recruitment.

Now, for the next big step – pay close attention here. Remember from our previous discussion that B_i is the number of individuals entering the population between time (*i*) and (*i*+1). If N_{i+1} is the number of individuals in the population at time (*i*+1), then B_i/N_{i+1} is the proportion (or probability) that an individual in the population at $i+1$ is one that entered the population between (*i*) and $(i+1)$. So, if (B_i/N_{i+1}) is the probability that an individual entered the population, then $1-(B_i/N_{i+1})$ is the probability that it was already in the population.

Does this sound familiar? It should. Think back – what is γ_i? It is the probability that if you're in the population at time (*i*) that you were also there at time (*i*-1), which is precisely the same thing! In other words,

$$\gamma_{i+1} = 1 - \frac{B_i}{N_{i+1}}$$

Thus, since

$$N_{i+1} = N_i \varphi_i + B_i$$

then we can write

$$\begin{aligned}\gamma_{i+1} &= 1 - \frac{B_i}{N_{i+1}} \\ &= 1 - \frac{(N_{i+1} - N_i\varphi_i)}{N_{i+1}} \\ &= \frac{N_i\varphi_i}{N_{i+1}} \\ &= \frac{\varphi_i}{\lambda_i}\end{aligned}$$

Or, re-arranging slightly,

$$\boxed{\lambda_i = \frac{\varphi_i}{\gamma_{i+1}}}$$

In other words, all we need are estimates of φ_i and γ_{i+1}, and we can derive estimates of λ_i, all without ever measuring population abundance (N)! Here we have a technique where, using simple mark-recapture data, we derive an explicit estimate of population growth, with considerable precision (as it turns out), without the need to estimate abundance.

begin sidebar

An alternative derivation

We can also derive this expression for λ using a slightly different approach. Let the size of the population at risk of capture (encounter) on occasions i and i+1 be N_i and N_{i+1}, respectively. A subset of these two sets (populations) of animals is the subset of all animals that are alive in the population at both times. At time i+1, the size of this set can be given as $N_i\varphi_i$. At time i, the size of that set can be given as $\gamma_{i+1}N_{i+1}$. Since both relationships give the size of the same set (population size alive at both times), then we can write

$$N_i\varphi_i = \gamma_{i+1}N_{i+1}$$

Since $\lambda = N_{i+1}/N_i$, then it follows clearly that $\lambda_i = \varphi_i/\gamma_{i+1}$, which is exactly the same relationship we derived above.

end sidebar

We also note that since $\lambda_i = \varphi_i + f_i$, then we can clearly also derive the following 2 expressions:

$$\gamma_{i+1} = \frac{\varphi_i}{\varphi_i + f_i}\left(= \frac{\varphi_i}{\lambda_i}\right) \qquad f_i = \varphi_i\left(\frac{1 - \gamma_{i+1}}{\gamma_{i+1}}\right)$$

This is the essence of the Pradel models in **MARK**. They rest on the duality between estimating φ_i going forward in time, and γ_i going backward in time. Moreover, because φ and γ can both be estimated in a general likelihood expression, not only can we estimate λ and f directly, but we can also derive estimates of the variance in both. Perhaps more importantly, we can address hypotheses about variation in one or more of these parameters using the standard linear models approach we've used throughout – constraining various parameters to be linear functions of one or more covariates.

13.3. Population λ versus Pradel's λ: are they equivalent?

At this point, you're probably asking yourself '*Is this too good to be true?*'. Well, like all things, there are assumptions, caveats, and conditions under which the Pradel models, and the parameters you can estimate with them, are 'meaningful'. First, and most importantly, the parameter λ estimated using the Pradel models* is a measure of the rate of change of the age class from which the encounter histories were derived - it is **not** necessarily a measure of the growth rate of the population! This is so important, we'll repeat it again, with emphasis:

> **The λ estimated from Pradel models is the realized growth rate of the *age class* from which the encounter histories were generated, *which is not necessarily (or even generally) equivalent to the growth rate of the population*...**

The λ estimated from the Pradel models *may* be a good measure if there is a single age class which contributes most of the variation in the growth of the population as a whole (or if you make somewhat heroic assumptions that the age structure of the population is stationary – in which case, growth of any individual age class is equivalent to the growth of the population as a whole).

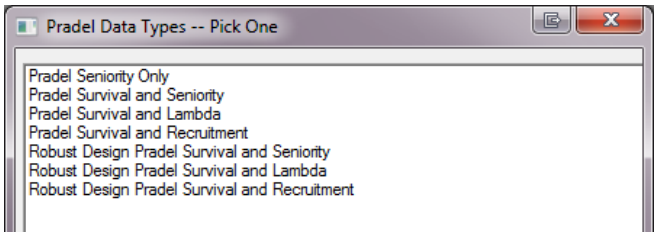
Second, the estimate of λ is only biologically meaningful if the study (sample) area stays constant. If you expand your study area, then both f and λ make little biological sense, because the population to which inferences are being made is also expanding or contracting. Further, even if the study area remains constant, some individuals can be missed in the first years of the study (when observers are learning their 'field craft'), and estimates of λ and f from early years may frequently be biased high. These methods also assume that all animals in the study area have some non-zero probability of being captured. Finally, significant trap response can lead to substantial bias.

But, these caveats notwithstanding, the Pradel models are potentially powerful tools for exploring population dynamics. We can look at variation in growth trajectory (λ) without the problems associated with abundance estimation.

13.4. Pradel models in MARK

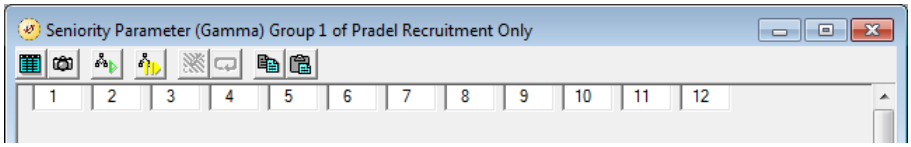
To demonstrate the Pradel models in **MARK**, we'll make use of the capsid data set (Muir, 1957), `pradel.inp` (this file is included when you install **MARK** - look in the `MARK/examples` subdirectory on your computer). Go ahead and start **MARK**, and begin a new project, reading in the capsid data file. There are 13 occasions, and 1 group. Now, once you've specified the input file, and entered the appropriate number of occasions and attribute groups, your next step is to specify which of the Pradel models you want to fit. When you click the '**`Pradel models including robust design`**' data type, you'll be presented with a pop-up window (shown at the top of the next page) listing 7 data types (4 primary for open populations, and 3 for models within a 'robust design' framework – see Chapter 15 for a full description of the robust design). For now, we'll focus on the (first) 4 'open population' models.

* It is worth noting that in his 1996 paper, Roger Pradel did not actually use the parameter λ, but instead used ρ to indicate the change in abundance between successive years. While using ρ for *realized* growth rate instead of λ eliminates confusion with use of λ as *projected* growth rate, **MARK** adopts the use of λ, since it is more commonly associated with measures of population growth.



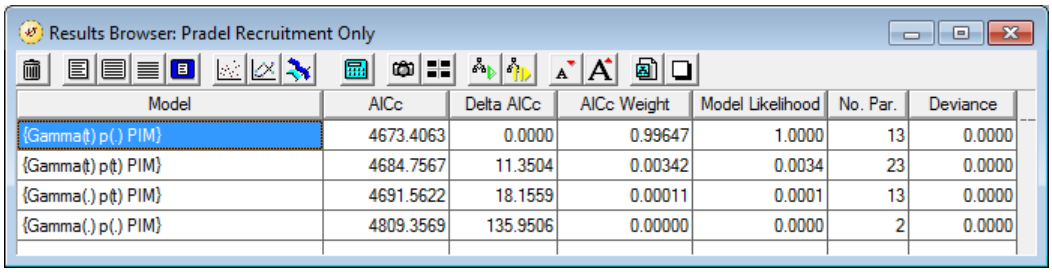
At this point, you need to decide which analysis you want to do. You could focus on estimation of the γ_i values alone (i.e., estimate the seniority probabilities). Estimation of γ can be useful in analysis of age-specific variation in 'recruitment to breeding state' (i.e., the transition from pre-breeder to breeder). The next 3 models consist of different pairs of parameters (for example, you could estimate survival and λ together by selecting the '**`Survival and Lambda`**' data type). You cannot estimate all of the various parameters simultaneously (i.e., you can't estimate φ, γ, f, and λ in one data type), since they are effectively a linear function of each other (such that estimating any two of them can provide estimates of the remaining parameters).

Let's run through all 4, starting with the '**`seniority only`**' data type. It also estimates the recapture parameter, p. Once you click the '**`OK`**' button, you'll immediately be presented with the PIM for the seniority parameter γ:



Now, your first puzzle - why only 1 row in the PIM? Why not the same sort of 'triangular' PIM we've seen for standard recapture models? The answer is because the Pradel models don't allow for 'age effects'. As noted by Franklin (2001), the reason for this is that the likelihood for estimating γ is conditioned on the entire encounter history, not just the portion following first capture as is the case when estimating φ under the CJS model. For example, **MARK** conditions on the full encounter history '`001101`' to estimate f and λ, whereas it conditions on only the '`1101`' portion to estimate φ and p. Therefore, age cannot be included because age cannot be estimated back to the initial zeros of the encounter history. Thus, you have no more than one row in the PIM (since each row corresponds to a different release cohort, and cohort and age are collinear). If age-specific likelihoods are desired, groups of animals can be created based on age.

The results browser (below) shows the results of fitting the 4 'standard' models to the capsid data.



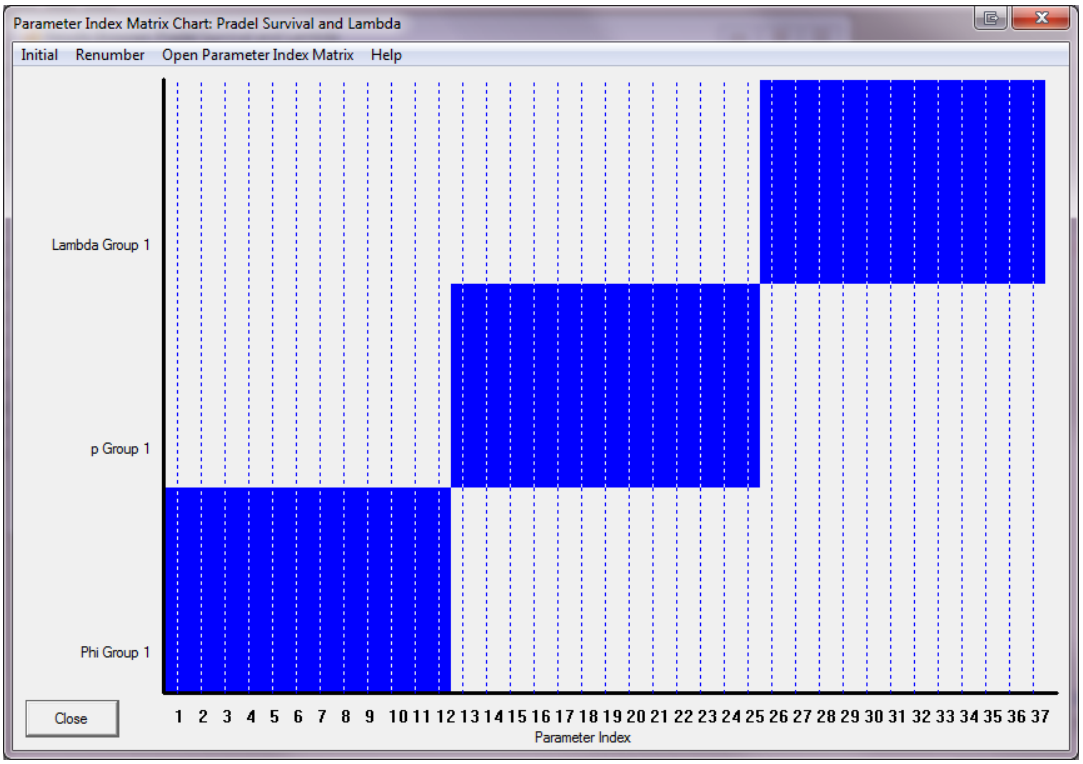
Model	AICc	Delta AICc	AICc Weight	Model Likelihood	No. Par.	Deviance
{Gamma(t) p(.) PIM}	4673.4063	0.0000	0.99647	1.0000	13	0.0000
{Gamma(t) p(t) PIM}	4684.7567	11.3504	0.00342	0.0034	23	0.0000
{Gamma(.) p(t) PIM}	4691.5622	18.1559	0.00011	0.0001	13	0.0000
{Gamma(.) p(.) PIM}	4809.3569	135.9506	0.00000	0.0000	2	0.0000

Based on these 4 models, and with a default $\hat{c}$ of 1.0, it appears as though a model with time-variation in γ and a constant encounter probability p is overwhelming supported by the data.

However, since we've mentioned $\hat{c}$, what about it? If you try to run a bootstrap, or median $\hat{c}$, **MARK** will quickly tell you that a neither of these GOF tests are available for the Pradel recruitment only model (and indeed, they aren't available for **any** of the Pradel models). At first glance, it might seem that you could simply flip the encounter histories back around the 'normal' forward direction (as if you were going to do a CJS analysis), and simply use the CJS bootstrap GOF test. However, this is inappropriate in general, since the likelihood is based on the full encounter history. Still, it 'may' be reasonable if your only interest is in estimating the γ parameters. For the moment, we can cautiously suggest that if all you're interested in is γ, and are using the Pradel 'seniority only' model, then since this model is identical to taking your encounter histories, flipping them, and running them through the CJS model, then the $\hat{c}$ from the CJS model may be appropriate. But – don't quote us (this is still a work in progress). For the other open population Pradel models ('**survival and seniority**', '**survival and lambda**', and '**survival and recruitment**'), this is likely to be incorrect.

These difficulties notwithstanding, it is likely these other models that will hold the greatest interest to you, since they provide information on parameters related to the growth of a population. We will focus in particular on the '**Survival and Lambda**' and '**Survival and Recruitment**' models.

Let's restart our analysis of the capsid data. This time, we will select the '**Survival and Lambda**' model. After we click the '**OK**' button, we are immediately presented with the open PIM for the apparent survival parameter *f*. If you open up the PIM chart (shown below), you'll see that there are 3 structural parameters involved in this model type: φ, p and λ.



If you look at each PIM separately, you'll see that each of them consists of 1 row (the reason for this has been discussed earlier). So, again, the modeling is relatively straightforward - you can apply constraints

quite simply to any one or more of the parameters – all three parameters are in the likelihood, and this, you can 'model' any of them (via the DM, for example).

begin sidebar

Pradel models and link functions

Several types of parameters have forced link functions, i.e., the link function is changed to the default value unless the user specifies '**Parm-Specific**' link functions. Specifically, the λ and f (recruitment) parameters of the Pradel data types are set to a *log* link function, even if the user selects the **sin**, **logit**, **loglog**, or **cloglog** link functions. Likewise, the population estimates ($\hat{N}$) in the (*i*) Jolly-Seber and POPAN and (*ii*) closed captures data types (discussed in Chapter 12 and Chapter 14, respectively) are also set to a log link function when the user selects the sin, logit, loglog, or cloglog link functions for the model. The reason for these changes from the user-specified link function for the model is that link functions that constrain these parameters to the [0, 1] interval will not work because the real parameters λ, f and N should not be in the [0, 1] interval.

Note: **MARK** will force the log link for λ, regardless of what other link function you select - but, there will be no indication of this in the output. So, for example, if you are evaluating the β_i values for a particular model, the β_i values are estimated on the log scale, so (i) if reconstituting by hand, you need to use the back-transform for the log link, and (ii) assessing if $\lambda > 0$ is equivalent to asking if $\beta_i > 0$.

In addition, there is at least one other situation involving Pradel models where you may need to pay particular attention to the link function. There is a logical necessity that $\varphi_i \leq \lambda_i$ for Pradel models where λ occurs as a structural parameter, i.e., is a parameter included in the likelihood. Consider for example, the situation where $f = 0$ (i.e., if there was no recruitment). In the absence of recruitment, the population would decline over time – the rate of the decline in a given year could not be less than φ_i, and thus $\varphi_i \leq \lambda_i$. You can enforce this constraint using the *cumulative logit* link function (introduced in Chapter 6, section 6.8.1). As you might recall, the cumulative logit link (CLogit) function is useful for constraining a set of parameters to monotonically increase.

Suppose for some parameter θ that you desire the relationship of $\hat{\theta}_1 \leq \hat{\theta}_2 \leq \hat{\theta}_3$, but do not want to enforce the relationship on the logit scale that

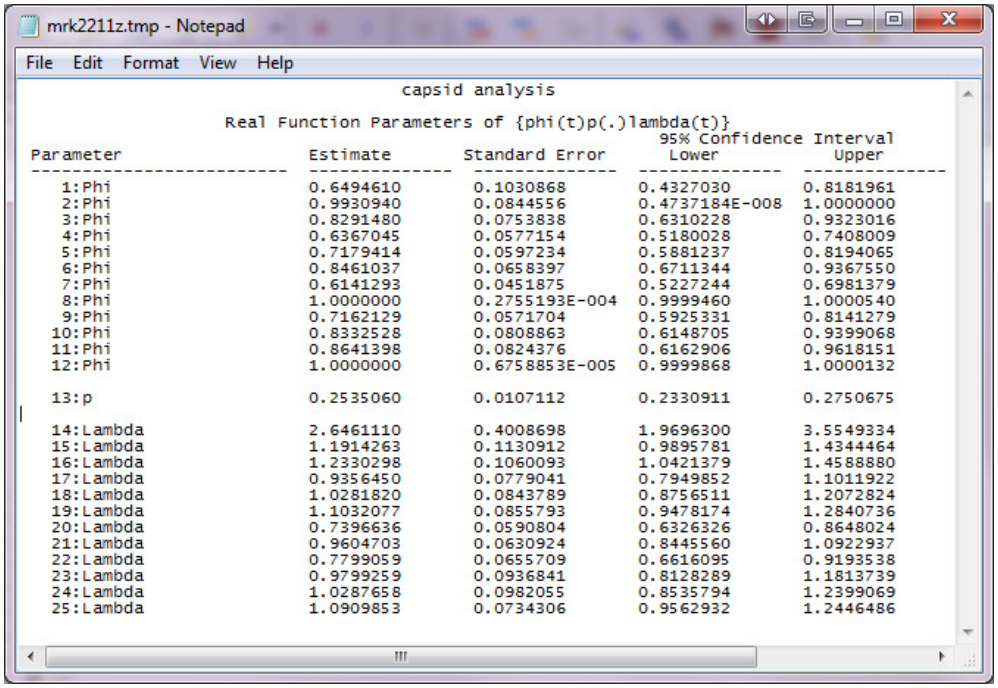
$$\text{logit}[\theta_2] - \text{logit}[\theta_1] = \text{logit}[\theta_3] - \text{logit}[\theta_2]$$

To use the CLogit link, you have to specify a separate CLogit link for each set of parameters that are to be constrained. In addition, you also have to specify the order of the parameters for the set. For the above example, the link function for each of the 3 parameters would be:

θ_1: `CLogit(1,1)`
θ_2: `CLogit(1,2)`
θ_3: `CLogit(1,3)`

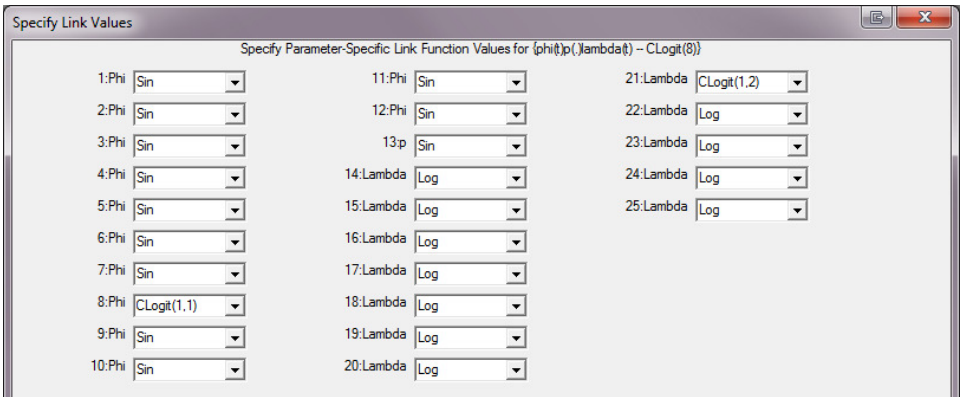
Consider the situation where you want to constrain $\varphi_i \leq \lambda_i$. We'll demonstrate the steps using the capsid data set introduced earlier (`contained in pradel.inp`). Re-start your analysis of these data, and fit model $\{\varphi_t p_\cdot \lambda_t\}$ to the data (i.e., the '**Pradel Survival and Lambda**' data type; time-dependence in φ and λ, but a constant 'dot' model for encounter probability p). We'll start using the default sin link for all parameters (remembering that in fact **MARK** will use the log link for λ). The parameter estimates are shown at the top of the next page.

While $\hat{\varphi}_i \leq \hat{\lambda}_i$ for most parameters, this is clearly not the case for for $\hat{\varphi}_8 = 1.0000 > \hat{\lambda}_8 = 0.9605$. Since this particular pair of parameters violates the logical necessity that $\hat{\varphi}_i \leq \hat{\lambda}_i$, we could simply re-run the model applying the CLogit link to these parameters only. To do this, we'll re-run model $\{\varphi_t p_\cdot \lambda_t\}$, except that this time we'll select the '**Parm-Specific**' link function option. Once you click the '**OK to Run**' button, you'll be presented with a popup window allowing you to specify the link function for a given parameter. Here, we're going to apply the CLogit link between φ_8 and λ_8. Recall that the CLogit link is specified by manually writing the word '`CLogit`' into the appropriate spot.



The Clogit function has 2 arguments: the first is the particular set (or pair) of parameters you want to apply the link to, and the second identifies the member of the pair. For example, `CLogit(1,1)` and `CLogit(1,2)` refer to the first pair of parameters, corresponding to φ_1 and λ_1, respectively. `CLogit(2,1)` and `CLogit(2,2)` refer to the second pair of parameters, corresponding to φ_2 and λ_2, respectively. And so on.

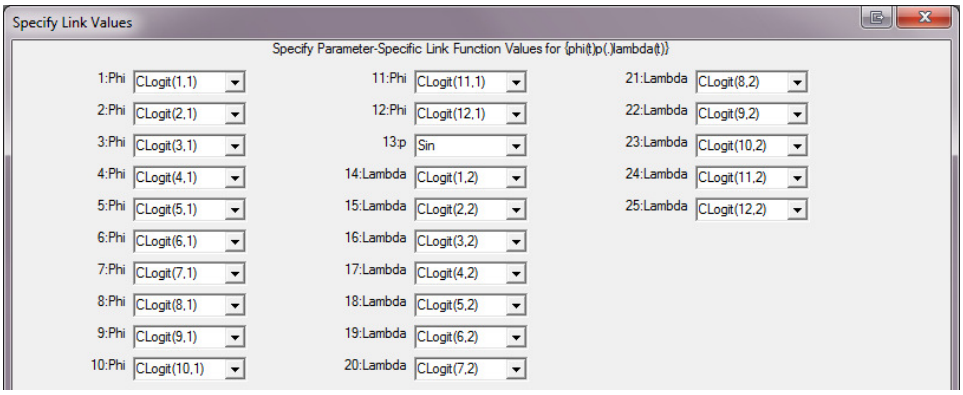
Here is the completed link function specification window:



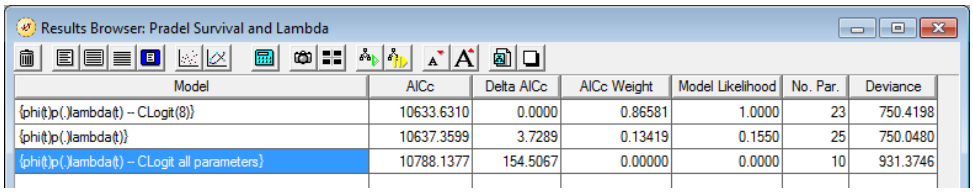
Note: if you are applying the CLogit link to only a subset of the φ and λ parameter pairs, you must remember to specify the log link function for the λ parameters you are not constraining using the CLogit. In this case, we specify the log link for parameters 14-20, and 22-25 (shown above). If we run the model with the CLogit link function applied to φ_8 and λ_8, we'll see that $\hat{\varphi}_8 = 0.9998 \leq \hat{\lambda}_8 = 0.9998$.

However, applying the CLogit only to those parameters which are identified (based on a first analysis) as violating the logical constraint that $\hat{\varphi}_i \leq \hat{\lambda}_i$ might appear rather *post hoc*. What about *a priori* applying a CLogit link to *all* of the φ and λ parameters? We'll see in a moment why this is **not** a good idea. For now, we'll plunge ahead as if it were.

Here is the completed link specification window if we apply the CLogit link function to all successive pairs of φ and λ parameters:

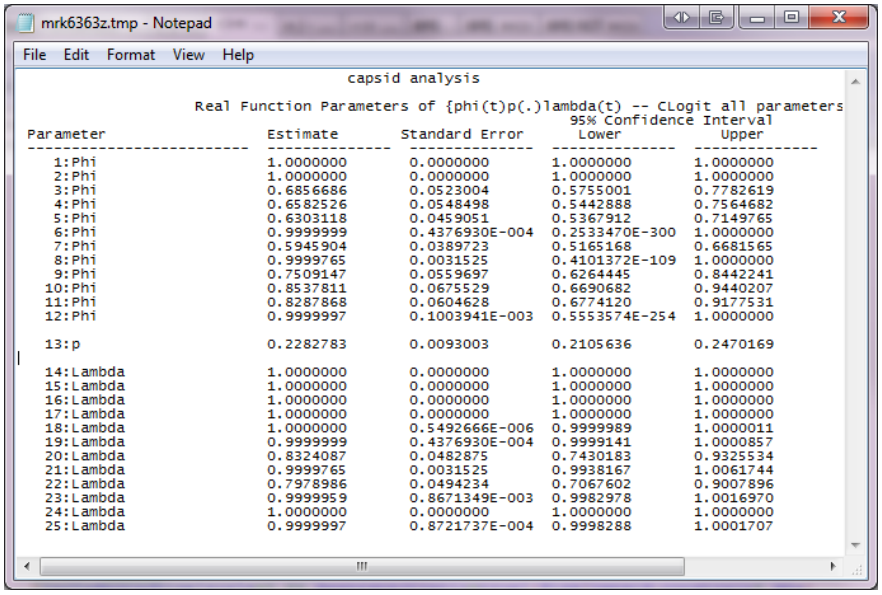


Pay attention to how successive pairs of parameters are indexed – the first argument in `CLogit(n,p)` is n=which pair, while p=which parameter in the pair. Click the '**OK**' button, and add the results to the browser.



Model	AICc	Delta AICc	AICc Weight	Model Likelihood	No. Par.	Deviance
{phi(t)p(.)lambda(t) -- CLogit(8)}	10633.6310	0.0000	0.86581	1.0000	23	750.4198
{phi(t)p(.)lambda(t)}	10637.3599	3.7289	0.13419	0.1550	25	750.0480
{phi(t)p(.)lambda(t) -- CLogit all parameters}	10788.1377	154.5067	0.00000	0.0000	10	931.3746

Hmmm. Based on the browser (below), you'll see that something has definitely 'gone wrong'. The first two models in the browser are the model with and without the CLogit link constraint applied to $[\varphi_8, \lambda_8]$, respectively. However, when we apply the CLogit link to all of the φ and λ parameters, we see that not only is the model deviance quite different (931.37 versus $\approx$ 750), but the number of estimated parameters is **much** lower (10 versus 23-25). If we look at the estimates,



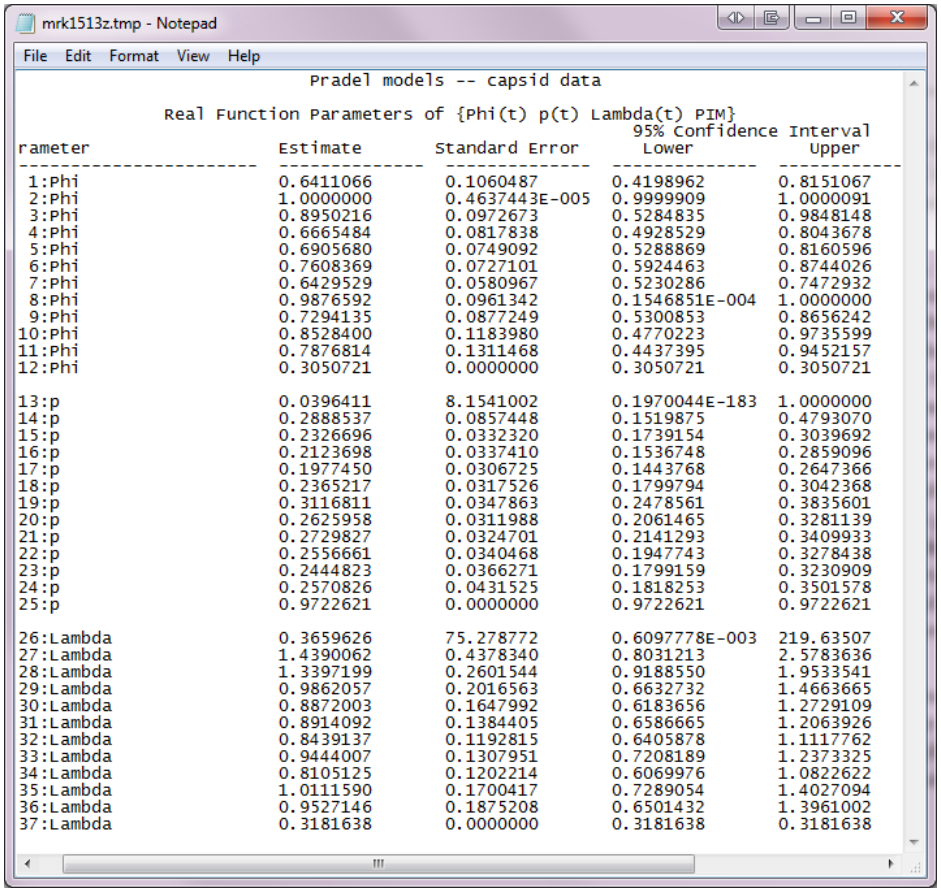
we see that the 'problem' is that many of the estimates of λ are estimated at the boundary, even though λ was previously estimated as > 1 for many of the estimates (see listing of estimates on p. 10).

What has happened? Well, the answer is somewhat explicit in the name of the link function: Clogit. The cumulative logit function is still a *logit* link, meaning, it constrains estimates to be bounded $[0, 1]$. While this isn't a problem for some parameters, it is clearly a problem for parameters such as λ, which are not upper-bounded at 1. So, you could/should apply the CLogit link function to enforce the constraint that $\varphi_i \leq \lambda_i$ only for those pairs of parameters where $\lambda_i \leq 1$.

end sidebar

So, at this point, this looks like perhaps the simplest thing we've done so far with **MARK**. However, there are several issues to consider. First, several parameters are confounded under the fully time-dependent model. For example, in model $\{\varphi_t p_t \lambda_t\}$ the first and last λ are inestimable because φ_1 is confounded with p_1 and φ_{k-1} is confounded with p_{k-1} (for k encounter occasions).

Here are reconstituted estimates from the fully time-dependent model $\{\varphi_t p_t \lambda_t\}$:


mrk1513z.tmp - Notepad

File Edit Format View Help

Pradel models -- capsid data

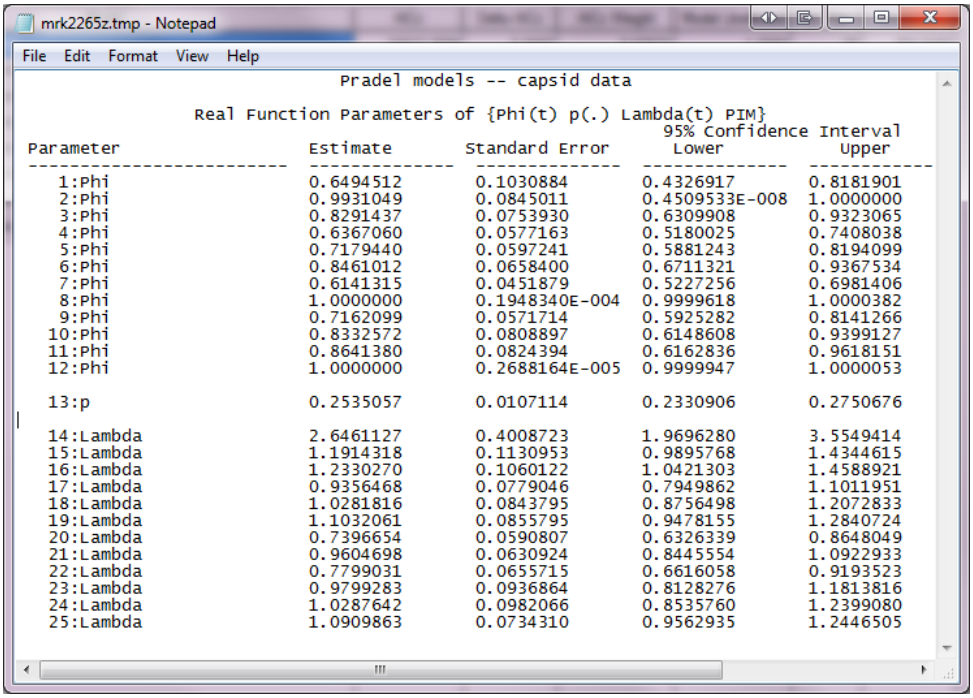
Real Function Parameters of {Phi(t) p(t) Lambda(t) PIM}

rameter	Estimate	Standard Error	95% Confidence Interval Lower	Upper
1:Phi	0.6411066	0.1060487	0.4198962	0.8151067
2:Phi	1.0000000	0.4637443E-005	0.9999909	1.0000091
3:Phi	0.8950216	0.0972673	0.5284835	0.9848148
4:Phi	0.6665484	0.0817838	0.4928529	0.8043678
5:Phi	0.6905680	0.0749092	0.5288869	0.8160596
6:Phi	0.7608369	0.0727101	0.5924463	0.8744026
7:Phi	0.6429529	0.0580967	0.5230286	0.7472932
8:Phi	0.9876592	0.0961342	0.1546851E-004	1.0000000
9:Phi	0.7294135	0.0877249	0.5300853	0.8656242
10:Phi	0.8528400	0.1183980	0.4770223	0.9735599
11:Phi	0.7876814	0.1311468	0.4437395	0.9452157
12:Phi	0.3050721	0.0000000	0.3050721	0.3050721
13:p	0.0396411	8.1541002	0.1970044E-183	1.0000000
14:p	0.2888537	0.0857448	0.1519875	0.4793070
15:p	0.2326696	0.0332320	0.1739154	0.3039692
16:p	0.2123698	0.0337410	0.1536748	0.2859096
17:p	0.1977450	0.0306725	0.1443768	0.2647366
18:p	0.2365217	0.0317526	0.1799794	0.3042368
19:p	0.3116811	0.0347863	0.2478561	0.3835601
20:p	0.2625958	0.0311988	0.2061465	0.3281139
21:p	0.2729827	0.0324701	0.2141293	0.3409933
22:p	0.2556661	0.0340468	0.1947743	0.3278438
23:p	0.2444823	0.0366271	0.1799159	0.3230909
24:p	0.2570826	0.0431525	0.1818253	0.3501578
25:p	0.9722621	0.0000000	0.9722621	0.9722621
26:Lambda	0.3659626	75.278772	0.6097778E-003	219.63507
27:Lambda	1.4390062	0.4378340	0.8031213	2.5783636
28:Lambda	1.3397199	0.2601544	0.9188550	1.9533541
29:Lambda	0.9862057	0.2016563	0.6632732	1.4663665
30:Lambda	0.8872003	0.1647992	0.6183656	1.2729109
31:Lambda	0.8914092	0.1384405	0.6586665	1.2063926
32:Lambda	0.8439137	0.1192815	0.6405878	1.1117762
33:Lambda	0.9444007	0.1307951	0.7208189	1.2373325
34:Lambda	0.8105125	0.1202214	0.6069976	1.0822622
35:Lambda	1.0111590	0.1700417	0.7289054	1.4027094
36:Lambda	0.9527146	0.1875208	0.6501432	1.3961002
37:Lambda	0.3181638	0.0000000	0.3181638	0.3181638

We see clearly that the first and last estimates of λ are 'problematic' (very large or impossibly small SE for both estimates). We also have the usual issues of inestimability of terminal survival and encounter parameters.

However, when constraints are placed on either φ or p, some of the variation in these parameters is taken up by λ, γ, or f, which often 'solves the estimability problem'. For example, if you look at the estimates from a model where the encounter probability p is constrained to be constant over time (e.g.,

model $\{\varphi_t p_. \lambda_t\}$), you will see that all of the estimates for λ appear reasonable.



mrk2265z.tmp - Notepad

File Edit Format View Help

Pradel models -- capsid data

Real Function Parameters of {Phi(t) p(.) Lambda(t) PIM}

Parameter	Estimate	Standard Error	95% Confidence Interval Lower	Upper
1:Phi	0.6494512	0.1030884	0.4326917	0.8181901
2:Phi	0.9931049	0.0845011	0.4509533E-008	1.0000000
3:Phi	0.8291437	0.0753930	0.6309908	0.9323065
4:Phi	0.6367060	0.0577163	0.5180025	0.7408038
5:Phi	0.7179440	0.0597241	0.5881243	0.8194099
6:Phi	0.8461012	0.0658400	0.6711321	0.9367534
7:Phi	0.6141315	0.0451879	0.5227256	0.6981406
8:Phi	1.0000000	0.1948340E-004	0.9999618	1.0000382
9:Phi	0.7162099	0.0571714	0.5925282	0.8141266
10:Phi	0.8332572	0.0808897	0.6148608	0.9399127
11:Phi	0.8641380	0.0824394	0.6162836	0.9618151
12:Phi	1.0000000	0.2688164E-005	0.9999947	1.0000053
13:p	0.2535057	0.0107114	0.2330906	0.2750676
14:Lambda	2.6461127	0.4008723	1.9696280	3.5549414
15:Lambda	1.1914318	0.1130953	0.9895768	1.4344615
16:Lambda	1.2330270	0.1060122	1.0421303	1.4588921
17:Lambda	0.9356468	0.0779046	0.7949862	1.1011951
18:Lambda	1.0281816	0.0843795	0.8756498	1.2072833
19:Lambda	1.1032061	0.0855795	0.9478155	1.2840724
20:Lambda	0.7396654	0.0590807	0.6326339	0.8648049
21:Lambda	0.9604698	0.0630924	0.8445554	1.0922933
22:Lambda	0.7799031	0.0655715	0.6616058	0.9193523
23:Lambda	0.9799283	0.0936864	0.8128276	1.1813816
24:Lambda	1.0287642	0.0982066	0.8535760	1.2399080
25:Lambda	1.0909863	0.0734310	0.9562935	1.2446505

Unfortunately, the issue of applying constraints to models where λ is a structural parameter is somewhat more complex than the preceding example suggests. For example, Franklin (2001) suggests that since λ, γ and f are (often) the parameters of biological interest in the Pradel models, that it is often best to model φ and p as completely time-dependent, and apply the constraints to λ, γ, or f.

While this might seem reasonable, there's a catch. If you do specify constraints on λ, γ, or f, you need to be a bit careful when interpreting the 'meaning' behind constraining these parameters. Since $\lambda_i = \varphi_i + f_i$, if a model is fit with time invariant (i.e., constant) λ, or where λ is constrained to follow a linear trend, but with time varying f, then this implies a direct inverse relationship between survival and recruitment (e.g., if λ is held constant, then if φ_i goes up, then f_i must go down). While this may be true in a general sense, it is doubtful that the link between the two operates on small time scales typically used in mark-recapture studies. As noted by Franklin (2001), models where φ is time invariant while λ is allowed to vary over time are (probably) reasonable, as variations in recruitment are the extra source of 'variation' in λ. More complex models involving covariates have the same difficulty. Population-level covariates (e.g., weather) are interpretable, but it is potentially difficult to interpret individual-based covariates as operating on population growth. The root of the problem is that while individual covariates could apply to survival probabilities, the recruitment parameter is not tied to any individual – it is a population-based, average recruitment per individual in the population. What is needed is a generalization of the general JS model where new entrants to a population are tied to existing members of the population, for example, if newborns were identified with their parents.*

So, as long as you take some care, then you'll be OK. There are always challenges in modeling parameters that are clearly linear functions of each other – be advised – think carefully. The Pradel models have great potential, for a number of applications - if you're careful, then you can apply them to a fairly broad set of problems. We introduce once such 'application' in the following section.

* An alternative approach based on random effects is discussed in Appendix D (section D.4.4).

13.5. an extension using the S and f parametrization...

Suppose you are interested on the relative degree to which recruitment or survival influence the dynamics of a population. For those of you with any background in matrix population models, this is an old concept: you have a metric describing the growth of the population (λ), and you want to determine the degree to which λ is 'sensitive' to variation in one or more elements of the demography of the population. For the matrix modelers among you, such a sensitivity analysis is usually expressed in terms of the rate of change of λ given a certain change in one of the matrix elements (a_{ij}). Done on a log scale, this 'sensitivity' analysis becomes an 'elasticity' analysis (the log scale expresses relative proportional contributions to λ). Now, in the typical 'sensitivity' or 'elasticity' analysis, the point is to determine what *would* happen to growth if a specified change is made in one or more vital rates. So, a *prospective* analysis. In the prospective context, sensitivity and elasticity are together referred to as 'perturbation' techniques, since the goal is to see how much projected growth would change as the system is 'perturbed' by changing one of the vital rates which contributes to population growth. There is a very large literature on the application of prospective perturbation techniques in conservation and population management.

However, in the *retrospective* context, the story is a little bit different. In this situation, we have an observed time series of λ_i, which we might estimate from our mark-encounter data. We want to know what the relative contributions of φ and f have been to the observed variation in λ. In other words, what *has* driven the basic pattern of variation in λ over time.

Several years ago, Jim Nichols and colleagues addressed this very question.* The approach they developed is (typically) very intuitive, and the result is rather elegant. The basic idea is that since γ_{i+1} is the probability that an individual in the population at time i+1 (and thus contributing to N_{i+1}) was also there at time i (and thus included in N_i), and since $N_{i+1} = N_i\varphi_i + B_i$, then

$$N_i\varphi_i = N_{i+1}\gamma_{i+1} \quad \text{and} \quad B_i = (1 - \gamma_{i+1})\, N_{i+1}$$

So, since $\lambda_i = N_{i+1}/N_i$, then

$$E(\lambda_i) = \frac{[\gamma_{i+1}N_{i+1} + (1 - \gamma_{i+1})N_{i+1}]}{E(N_i)}$$

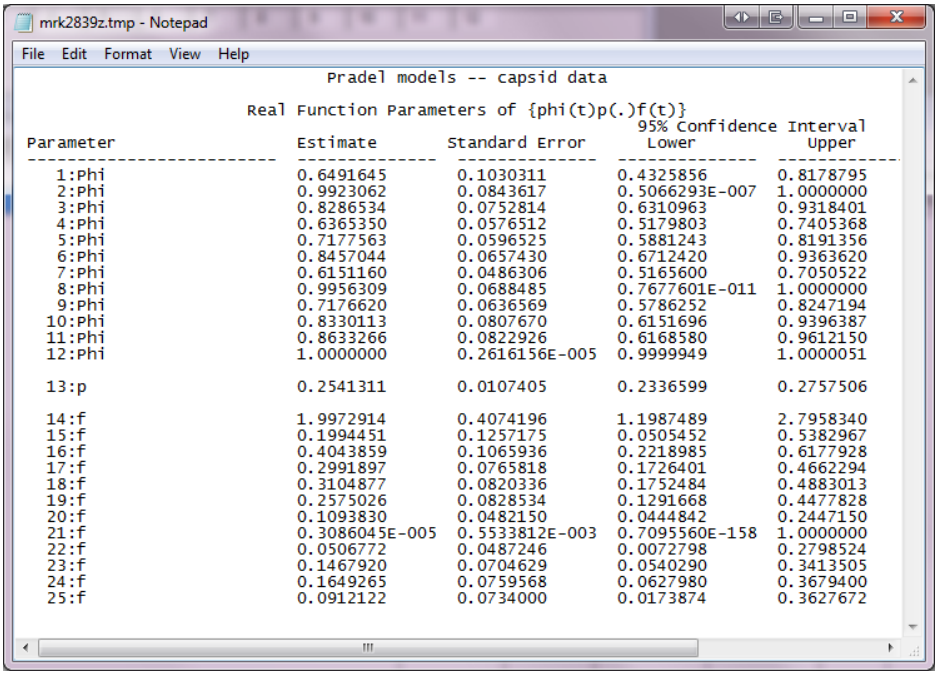
Since the abundance term, N_{i+1}, is the same for both product terms in the numerator, then the γ_{i+1} in the first term in the numerator is interpretable as the contribution of survivors from time i to time i+1, while the $(1 - \gamma_{i+1})$ in the second term of the numerator is the contribution of new recruits into the population. So, the two terms give the relative contributions of survivors and new recruits to λ, which is conceptually analogous to the elasticities of both terms. The details (and several very clever extensions to multi-state and robust design models) are discussed at length in the Nichols paper.

However, this approach involves 'doing algebra' with estimates of γ only. A more direct, and ultimately more flexible approach is to re-parameterize the likelihood in terms of survival and recruitment directly. This is the basis of the '**`Survival and Recruitment`**' model. Re-start the analysis of the capsid data set, this time selecting the '**`Survival and Recruitment`**' model. If you look at the PIM chart, you'll see that there are only 3 structural parameters: φ, p and f. For this model type, the realized growth λ is estimated as a *derived parameter* (meaning, it is derived 'by algebra', outside the likelihood). This has one immediate implication, which refers back to our earlier discussion of applying constraints to Pradel models. Because λ is a derived parameter (not in the likelihood), you cannot put a constraint on

* Nichols, J. D., J. E. Hines, J-D. Lebreton, R. Pradel. (2000) Estimation of contributions to population growth: a reverse-time capture-recapture approach. *Ecology*, **81**, 3362-3376.

λ. Rather, you might apply constraints to the processes which contribute to λ (i.e., S and f).

Let's proceed by fitting a single model to the capsid data: model $\{\varphi_t p f_t\}$. We'll use constant encounter probabilities over time to eliminate some of the confounding problems inherent in fully time-dependent models. Here are the reconstituted parameter estimates from this model


mrk2839z.tmp - Notepad

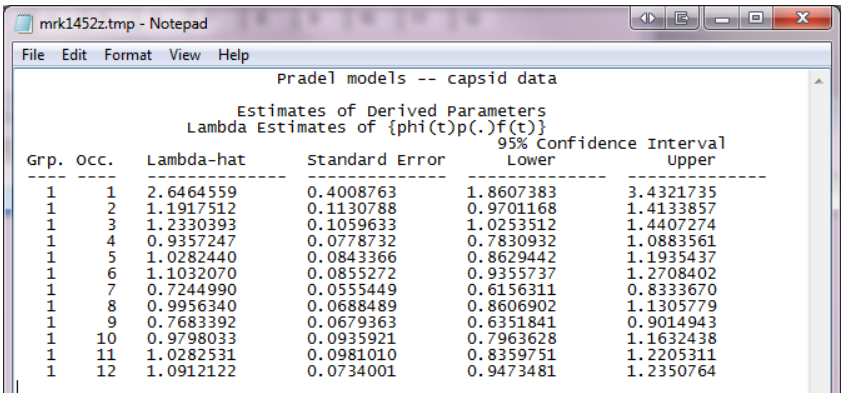
File Edit Format View Help

Pradel models -- capsid data

Real Function Parameters of {phi(t)p(.)f(t)}

Parameter	Estimate	Standard Error	95% Confidence Interval Lower	Upper
1:Phi	0.6491645	0.1030311	0.4325856	0.8178795
2:Phi	0.9923062	0.0843617	0.5066293E-007	1.0000000
3:Phi	0.8286534	0.0752814	0.6310963	0.9318401
4:Phi	0.6365350	0.0576512	0.5179803	0.7405368
5:Phi	0.7177563	0.0596525	0.5881243	0.8191356
6:Phi	0.8457044	0.0657430	0.6712420	0.9363620
7:Phi	0.6151160	0.0486306	0.5165600	0.7050522
8:Phi	0.9956309	0.0688485	0.7677601E-011	1.0000000
9:Phi	0.7176620	0.0636569	0.5786252	0.8247194
10:Phi	0.8330113	0.0807670	0.6151696	0.9396387
11:Phi	0.8633266	0.0822926	0.6168580	0.9612150
12:Phi	1.0000000	0.2616156E-005	0.9999949	1.0000051
13:p	0.2541311	0.0107405	0.2336599	0.2757506
14:f	1.9972914	0.4074196	1.1987489	2.7958340
15:f	0.1994451	0.1257175	0.0505452	0.5382967
16:f	0.4043859	0.1065936	0.2218985	0.6177928
17:f	0.2991897	0.0765818	0.1726401	0.4662294
18:f	0.3104877	0.0820336	0.1752484	0.4883013
19:f	0.2575026	0.0828534	0.1291668	0.4477828
20:f	0.1093830	0.0482150	0.0444842	0.2447150
21:f	0.3086045E-005	0.5533812E-003	0.7095560E-158	1.0000000
22:f	0.0506772	0.0487246	0.0072798	0.2798524
23:f	0.1467920	0.0704629	0.0540290	0.3413505
24:f	0.1649265	0.0759568	0.0627980	0.3679400
25:f	0.0912122	0.0734000	0.0173874	0.3627672

and the reconstituted estimates of λ (on the real scale):


mrk1452z.tmp - Notepad

File Edit Format View Help

Pradel models -- capsid data

Estimates of Derived Parameters
Lambda Estimates of {phi(t)p(.)f(t)}

Grp.	Occ.	Lambda-hat	Standard Error	95% Confidence Interval Lower	Upper
1	1	2.6464559	0.4008763	1.8607383	3.4321735
1	2	1.1917512	0.1130788	0.9701168	1.4133857
1	3	1.2330393	0.1059633	1.0253512	1.4407274
1	4	0.9357247	0.0778732	0.7830932	1.0883561
1	5	1.0282440	0.0843366	0.8629442	1.1935437
1	6	1.1032070	0.0855272	0.9355737	1.2708402
1	7	0.7244990	0.0555449	0.6156311	0.8333670
1	8	0.9956340	0.0688489	0.8606902	1.1305779
1	9	0.7683392	0.0679363	0.6351841	0.9014943
1	10	0.9798033	0.0935921	0.7963628	1.1632438
1	11	1.0282531	0.0981010	0.8359751	1.2205311
1	12	1.0912122	0.0734001	0.9473481	1.2350764

To reinforce the relationship between survival, recruitment, and λ, compare (i) the derived estimate of λ provided by **MARK** (above), and (ii) the value of the sum of the estimates of survival and recruitment (shown at the bottom of the preceding page).

Since

$$\lambda_i = \varphi_i + \frac{B_i}{N_i} = \varphi_i + f_i$$

then (i) and (ii) should be identical, as they are shown to be (to within rounding error) in the following table for a sub-sample of intervals:

interval	*derived* $\hat{\lambda}$	$\hat{\varphi}$	$\hat{f}$	$\hat{\varphi}+\hat{f}$
2	1.1918	0.9923	0.1994	1.1918
3	1.2330	0.8287	0.4044	1.2330
4	0.9357	0.6365	0.2992	0.9357
5	1.0282	0.7178	0.3105	1.0282
6	1.1032	0.8457	0.2575	1.1032
7	0.7245	0.6151	0.1094	0.7245

We could express the proportional contribution of (say) survival φ to realized growth λ simply as $\varphi/(\varphi+f)$. The variance of this proportion can be estimated using the Delta method (Appendix B). *

An important point to make here is that interpreted as elasticities in this fashion, the approach Nichols and colleagues describe partitions variation in realized λ due to contributions from survivors and recruits, without the (somewhat) limiting assumptions of some matrix models approach (there is a growing literature debating whether or not some of the ergodic assumptions underlying matrix methods really are a limit, but that isn't something we want to get into here).

One final point: note that all we've done up until now is talk about net 'additions' and 'subtractions'. We haven't partitioned these any further. For example, we haven't partitioned additions into '*in situ* recruits' and 'immigrants'. We may, in fact, not be satisfied with simply using a 'summary accounting' like 'total recruits' or 'total subtractions' - we may want to know how many of each are due to underlying, lower-level processes (like births, immigration, deaths, or emigration). However, to do that, we'd need to consider different approaches: the Jolly-Seber (and related) models (introduced in Chapter 12), and the robust design (which we introduce in Chapter 15). But, if partitioning λ into summary contributions of total recruits and total losses is what you're after, then the Pradel models may be of some use.

begin sidebar

λ = 'survival' + 'recruitment': be careful!

Another example of a potential pitfall. In the preceding, we made use of the fact that realized λ can be estimated as the sum of survival and per capita recruitment, both estimated over a given interval. However, you need to be careful in how you are 'handling' recruitment. Consider, for example, a population with 2 age classes: babies, and adults.

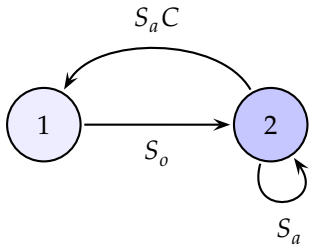
Assume that adults start breeding at age 2 years (i.e., they don't breed as yearlings), and on average produce C babies. Babies survive with probability S_o, and thus become adults, which survive with probability S_a.

Assuming the population is censused after breeding, the population can be described by the life-cycle diagram shown on the next page, where node 1 refers to babies, and node 2 refers to adults (age 1 yr and older). The self-loop on node 2 indicates that survival does not vary with age among adults. The fertility arc (connecting node 2 back to node 1) represents the expected contribution of each individual in node 2 at time (i) to the baby age class at time (i+1).

* If we let $\rho = \varphi/(\varphi+f)$, then

$$\widehat{\mathrm{var}}(\hat{\rho}) = \begin{bmatrix} \left(\frac{\hat{f}}{(\hat{\varphi}+\hat{f})^2}\right) & \left(-\frac{\hat{\varphi}}{(\hat{\varphi}+\hat{f})^2}\right) \end{bmatrix} \begin{bmatrix} \widehat{\mathrm{var}}(\hat{\varphi}) & \widehat{\mathrm{cov}}(\hat{\varphi},\hat{f}) \\ \widehat{\mathrm{cov}}(\hat{f},\hat{\varphi}) & \widehat{\mathrm{var}}(\hat{f}) \end{bmatrix} \begin{bmatrix} \frac{\hat{f}}{(\hat{\varphi}+\hat{f})^2} \\ -\frac{\hat{\varphi}}{(\hat{\varphi}+\hat{f})^2} \end{bmatrix}$$

where the variance and covariance of $\hat{\varphi}$ and $\hat{f}$ can be output from **MARK** (see Appendix B).



From the life cycle graph (above), we can derive the corresponding projection matrix

$$\mathbf{A} = \begin{pmatrix} 0 & S_aC \\ S_o & S_a \end{pmatrix}$$

Assume that $C = 0.42$, $S_a = 0.9$, and $S_o = 0.5$. Thus, the projection matrix **A** is

$$\mathbf{A} = \begin{pmatrix} 0 & S_aC \\ S_o & S_a \end{pmatrix} = \begin{pmatrix} 0.000 & 0.378 \\ 0.500 & 0.900 \end{pmatrix}$$

from which we can determine that projected $\lambda = 1.0757$. [The use of life cycle diagrams, projection matrices, and various metrics extracted from such matrices, is discussed in most modern texts on population biology – Caswell (2001) is the standard reference].

OK, but what if you had used a different approach, based on the logic underlying the derivation of the Pradel models? In other words, $\lambda_i = \varphi_i + f_i$. Assume we know that $(\varphi_i =) S_a = 0.90$. That would appear to be half of our equation for λ. What about recruitment, f_i? Well, for the Pradel models, we're interested in recruitment to the adult age class - the number of individuals entering the adult population between (i) and (i+1) for each individual adult at (i). Well, if you stare at the life-cycle diagram on the preceding page, it might seem to be obvious that recruitment is simply the number of babies who become adults. True, but how many babies are there? Recall that we're estimating growth rate λ without having estimates of abundance. Well, as a first stab at the answer, you might think that the number of babies surviving to adulthood is a function of how many babies are produced by current adults (which is $S_a.C.S_o$; because this is a post-breeding census, you pre-multiply by S_a since a current adult has to survive in order to produce babies next year - this is indicated by the product $S_a.C$ on the fertility arc connecting node 2 back to node 1 in the life-cycle diagram). So, you might try to estimate λ as $\lambda = S_a + S_a.C.S_o = 0.9 + 0.189 = 1.089$.

Unfortunately, this estimate (1.089) is not the same as the 'true' estimate of projected growth rate derived from the projection matrix (1.0757). Why the difference? The difference is due to the fact that recruitment between (i) and (i+1) is a function of how many babies there were at time (i). The product $S_a.C.S_o$ gives the projected recruitment between (i+1) and (i+2)! Why? Look carefully - the product $S_a.C.S_o$ covers two time intervals: one for current adults (S_a), and one for babies produced next year by those adults (S_o).

So, how would you solve this problem? Fairly easily – you simply need to remember that for an exponentially growing population, ΔN for any age class over (t) time intervals is simply $N\lambda^t$. Similarly, since the projection of an exponentially growing population is time-symmetric, you could also project backwards, and say that the size of the population (t) time units in past is simply $N\lambda^{-t}$. Which is important...because?? It's important because you want to know how many of babies at time (i) will recruit (become adults) between (i) and (i+1). Since the product $S_a.C.S_o$ in fact gives recruitment 2 time steps ahead, what you need to do is 'back-step' this product by 1 time step, which (as noted) is given simply by λ^{-1}. For our numerical example, where λ (from the matrix) is given as 1.0757, then $(1.0757)^{-1} = 0.92963$. So, we correct (or 'back-step') our recruitment term as $(0.92963).S_a.C.S_o = 0.1757$. Thus, $\lambda = 0.9 + 0.1757 = 1.0757$, which is exactly the same value we got from the projection matrix.

OK – admittedly a somewhat 'artificial' problem, but remember: although the basic logic underlying

the temporal symmetry approach to estimating λ is relatively simple, you do need to pay close attention to what is going on (which we suppose is a general truism for most things, but we'll make the point again here).

end sidebar

13.6. Pradel models and Jolly-Seber estimation

We began this chapter by noting that population dynamics in the broad sense is determined by the net balance of 'additions' and 'subtractions'. The Pradel models, which we've introduced in this chapter, are one of several approaches available in **MARK** for partitioning one or more components of the dynamics of a population.

However, there are some important differences between Pradel models and the classical approaches generally referred to as *Jolly-Seber* models, introduced earlier in Chapter 12. In fact, the Pradel models considered in this chapter, are in effect a special (conditional) case of the more general Jolly-Seber model (as discussed in Chapter 12). One important difference is that the Pradel models, and the Cormack-Jolly-Seber (CJS) models we've considered in detail for analysis of live encounter data, condition on events since marking, and do not explicitly try to model events prior to the first encounter.

Another major difference is that neither the Pradel or CJS models specifically model abundance. In many cases, however, estimating abundance in open populations is important. For that, you need to consider Jolly-Seber and related models (Chapter 12).

13.7. Summary

That's the end of our very quick stroll through the Pradel models. We've seen how a simple 'flip' of the encounter history can yield all sorts of interesting information on the processes underlying the dynamics of our population. We can estimate population growth, without the 'messy' job of estimating abundance. Moreover, we can partition variation in population growth due to relative contributions of recruitment and mortality. Pretty neat stuff. But, we wouldn't want to presume that estimating abundance, and all of the other elements which contribute to the dynamics of a population, are not important - we'll deal with this in the next chapter.

CHAPTER 14

Closed population capture-recapture models

Paul Lukacs, *University of Montana*

A fair argument could be made that the marking of individuals in a wild population was originally motivated by the desire to estimate a fundamental parameter: abundance (i.e., population size). By comparing the relative proportions of marked and unmarked animals in successive samples, various estimators of animal abundance could be derived. In this chapter, we consider the theory and mechanics of estimation of abundance from *closed* population capture-recapture data, using program **MARK**.* Here, the population of interest is assumed to be closed geographically – no movement on or off the study area – and demographically – no births or deaths.

14.1. The basic idea

How many individuals are there in the sampled population? Well, if the population is (or assumed to be) closed, then the number of individuals in the population being sampled is a constant over time. Meaning, the population size does not change at each sampling event. With a little thought, you quickly realize that the canonical estimate of population size is a function of (i) how many unique individuals are encountered over all sampling events, and (ii) what the probability is of encountering an individual at least once. For a single sampling event, we can express this more formally as

$$\hat{N} = \frac{n}{\hat{p}}$$

where the numerator (n) is the number of unique individuals encountered, and the denominator (p) is the probability that any individual will be encountered.

This expression makes good intuitive sense. For example, suppose that you capture 50 individuals ($n = 50$), and the encounter probability is $p = 0.5$, then clearly, since there is a 50:50 cHance that you will miss an individual instead of encountering it, then

$$\begin{aligned}\hat{N} &= \frac{n}{\hat{p}} \\ &= \frac{50}{0.5} = 100\end{aligned}$$

* All of the likelihood-based models from Program **CAPTURE** can be built in **MARK** plus numerous models that have been developed since then. Program **MARK** supersedes program **CAPTURE** in all important respects.

© Cooch & White (2014)

14.1.1. The Lincoln-Petersen estimator - a quick review

In the preceding, we assume that we have an estimate of the encounter probability p. The most general approach to estimating abundance, and p, in closed populations is based on what is known as the Lincoln-Petersen estimator (hereafter, the 'LP estimator'). The LP estimator is appropriate when there are just two sampling occasions, and the population is closed between the two occasions. Imagine you go out on the first occasion, capture a sample of individuals from the population you are interested in, mark and release them back into the population. Then, on the second occasion, you re-sample from (what you hope is) the same population. In this second sample, there will (potentially) be two types of individuals: those that are unmarked (not previously captured) and those with marks (individuals captured and marked on the first occasion).

For such a study, there are only 4 possible encounter histories: '11' (captured, marked and released on occasion 1, recaptured on occasion 2), '10' (captured, marked and released on occasion 1, not recaptured on occasion 2), '01' (not captured on occasion 1, captured, marked and released for the first time on occasion 2), and '00' (not captured at either occasion 1 or occasion 2). Clearly, the number of individuals with encounter history '00' is not known directly, but must be estimated. So, the estimation of abundance proceeds by using the number of individuals observed who were encountered at least once.

Following convention, let x_{11} be the number of individuals caught on both occasions, x_{10} be the number of individuals caught on only the first occasion, and x_{01} be the number of individuals caught only on the second occasion. Let $n_1 = x_{11} + x_{10}$ be the total number of individuals caught on the first occasion, and $n_2 = x_{11} + x_{01}$ be the total number of individuals caught on the second occasion. Finally, let $m_2 = x_{11}$ be the number of individuals caught on both occasions. Thus, the number of distinct individuals captured during the study is $r = n_1 + n_2 - m_2$ (make sure you understand the algebra).

Under some assumptions (i.e., (i) the population is closed, (ii) marks are not lost or overlooked, and (iii) all animals are equally likely to be captured, regardless of whether or not they've been previously captured), we can proceed with the estimation of abundance. The most intuitive (and easily most familiar) approach to estimating abundance using an LP estimator is to note that the proportion of marked animals in the population after the first sample is n_1/N, where N is the size of the population (which, of course, is what we're trying to estimate). Note that the numerator of this expression (n_1) is known. If our assumption that all individuals (marked or not) are equally catchable, then this proportion should be equivalent to the proportion of marked individuals in the second sample. In other words,

$$\frac{n_1}{N} = \frac{m_2}{n_2}$$

A little algebra, and we come up with the familiar LP estimator for abundance, as

$$\hat{N} = \frac{n_1 n_2}{m_2}$$

We might also use the canonical form noted earlier, estimating $\hat{p} = m_2/n_2$.

Thus

$$\begin{aligned}\hat{N} &= \frac{n_1}{\hat{p}_1} \\ &= \frac{n_1 n_2}{m_2}\end{aligned}$$

which is identical to the 'intuitive' LP estimator we derived earlier.

14.2. Likelihood

While the 'algebraic' estimator for N developed in the preceding section is simple, reasonably intuitive and undoubtedly quite familiar, here we consider a more formal approach.

14.2.1. full likelihood approach

We start by expressing the probability distribution for the two sample study introduced in the preceding section as

$$P(n_1, n_2, m_2 \mid N, p_1, p_2) = \frac{N!}{m_2!(n_1 - m_1)!(n_2 - m_2)!(N - r)!} \times (p_1 p_2)^{m_2} [p_1(1 - p_2)]^{(n_1 - m_2)} [(1 - p_1)p_2]^{(n_2 - m_2)} [(1 - p_1)(1 - p_2)]^{(N-r)}$$

Two important things to note, here. First, N appears in the multinomial coefficient of the likelihood function. Second, the probability expression is written including a term for each encounter history, and with the exponent representing the number of individuals with a given encounter history (expressed in the standard notation introduced earlier). For example, the probability of encounter history '11' is $p_1 p_2$, the probability of encounter history '10' is $p_1(1 - p_2)$, and so on. But, note also that the encounter history representing individuals that were never caught (i.e., '00' for a two occasion case) also appears in the likelihood (but not in the data – i.e., the encounter histories file – since (obviously) there are no data for individuals that were never captured!).

More generally, we can write the likelihood as

$$\mathcal{L}(N, \mathbf{p} \mid \text{data}) \propto \frac{N!}{(N - M_{t+1})!} \prod_h P[h]^{n_h} \cdot P[\text{not encountered}]^{N - M_{t+1}}$$

where $\mathbf{p}$ is the vector of encounter probability parameters, M_{t+1} is the number of unique animals marked, and n_h is the number (frequency) of individuals with encounter history h.

Now, it is possible to rewrite the likelihood in terms of the number of individuals never caught, f_0, such that $f_0 = N - M_{t+1}$ (the notation 'f_0' originates from the frequency (count) of animals observed 0 times). The likelihood now becomes

$$\mathcal{L}(f_0, \mathbf{p} \mid \text{data}) \propto \frac{(f_0 + M_{t+1})!}{f_0!} \prod_h P[h]^{n_h} \cdot P[\text{not encountered}]^{f_0}.$$

The f_0 parametrization is useful computationally because f_0 is bounded on the interval $[0, \infty]$, thus forcing the logical constraint that $\hat{N} \geq M_{t+1}$. In fact, **MARK** uses the f_0 parametrization for ease of computation by using the log link function to constrain $\hat{f}_0 \geq 0$, but presents the results in terms of N as a *derived* parameter (i.e., $\hat{N} = \hat{f}_0 + M_{t+1}$ and $\widehat{\text{var}}[\hat{N}] = \widehat{\text{var}}[\hat{f}_0]$).

The fact that **MARK** uses f_0, the number of individuals never caught, in the likelihood has important implications you must keep in mind. Consider a study with two different locations – you may be interested as to whether or not there is a difference between locations in abundance. How would you build a model where (say) you set $N_1 = N_2$? Answer – you can't. You can only apply constraints to parameters that are included in the likelihood. Abundance N isn't in the likelihood, so you can't build models that constrain N.

OK, fine, but can't we constrain f_0, which *is* in the likelihood? At first glance, this might seem reasonable. Since $\hat{N} = \hat{f}_0 + M_{t+1}$, and since M_{t+1} is a constant, then $\hat{N} \propto \hat{f}_0$. So isn't constraining $\hat{f}_0$ effectively equivalent to constraining $\hat{N}$? if you think about it for a moment, you should realize than answers is 'no, this is not reasonable'. Consider setting $\hat{f}_{0,1} = \hat{f}_{0,2}$. This is easy enough to do in **MARK**, but, does it really make sense to say that the number never caught is the same in the 2 locations? Probably not. So, in short, you cannot constrain N in any meaningful way.

14.2.2. conditional likelihood

It is sometimes convenient to use a conditional likelihood approach to estimating abundance, where N is not actually considered as a parameter. This is possible if you condition the analysis only on those individuals which are encountered (i.e., r). The probability that any individual in the population is encountered at least once during a two-sample study is

$$p^* = 1.0 - (1 - p_1)(1 - p_2)$$

Thus, we can re-write the conditional probability expression for the capture histories as

$$P(\{x_{ij}\} \mid r, p_1, p_2) = \frac{r!}{x_{11}!x_{10}!x_{01}!} \times \left(\frac{p_1 p_2}{p^*}\right)^{x_{11}} \left(\frac{p_1(1-p_2)}{p^*}\right)^{x_{10}} \left(\frac{(1-p_1)p_2}{p^*}\right)^{x_{01}}$$

The ML estimates for this model are again fairly easy to derive (see Williams, Nichols & Conroy 2002 for the details).

The primary advantage of using this conditional likelihood approach is that individual covariates can be used to model the encounter process. Individual covariates cannot be used with the full likelihood approach introduced in the preceding section, because the term $(1 - p_1)(1 - p_2)...(1 - p_t)$ is included in the likelihood, and no covariate value is available for animals that were never captured. In contrast, the unconditional likelihood approach conditions this multinomial term out of the likelihood, and so an individual covariate can be measured for each of the animals included in the likelihood. When individual covariates are used, a Horvitz-Thompson estimator is used to estimate N:

$$\hat{N} = \sum_{i=1}^{M_{t+1}} \frac{1}{1 - [1 - \hat{p}_1(x_i)][1 - \hat{p}_2(x_i)]...[1 - \hat{p}_t(x_i)]}$$

An example is perhaps the best way to illustrate the difference between the full and conditional likelihood approaches. Consider the 4 possible encounter histories for 2 sampling occasions:

encounter history	*probability*
11	$p_1 p_2$
10	$p_1(1 - p_2)$
01	$(1 - p_1)p_2$
00	$(1 - p_1)(1 - p_2)$

For each of the encounter histories except the last, the number of animals with the specific encounter history is known. For the last encounter history, the number of animals is $f_0 = (N - M_{t+1})$, i.e., the

population size (N) minus the number of animals known to have been in the population (M_{t+1}). The approach (first described by Huggins 1989, 1991) was to condition this last encounter history out of the likelihood by dividing the quantity '1 minus this last history' into each of the others. The result is a new multinomial distribution that still sums to one. The derived parameter N is then estimated as $M_{t+1}/[1-(1-\hat{p})(1-\hat{p})(1-\hat{p})]$ for data with no individual covariates. A more complex estimator is required for models that include individual covariates to model the p parameters.

Here's a simple example of how this works, given 2 occasions. Let $p_1 = 0.4$, $p_2 = 0.3$. In the following, we tabulate both the unconditional probability of a given encounter history (i.e., where N is in the likelihood), and the conditional probability of the encounter history, where the individuals not seen are not included (i.e., are 'conditioned out'). Note that if $p_1 = 0.4$ and $p_2 = 0.3$, then the probability of not being captured at all is $(1-p_1)(1-p_2) = 0.42$, such that the probability of being captured at least once is `Pr(capt)`=$p^* = (1-0.42) = 0.58$.

history	*unconditional Pr(history)*		*Pr(history | captured)*	
`11`	$p_1 p_2$	$(0.4 \times 0.3) = 0.12$	$(p_1 p_2)/p^*$	$0.12/0.58 = 0.207$
`10`	$p_1(1-p_2)$	$0.4(1-0.3) = 0.28$	$[p_1(1-p_2)]/p^*$	$0.28/0.58 = 0.483$
`01`	$(1-p_1)p_2$	$(1-0.4)0.3 = 0.18$	$[(1-p_1)p_2]/p^*$	$0.18/0.58 = 0.310$
`00`	$(1-p_1)(1-p_2)$	$(1-0.4)(1-0.3) = 0.42$	(not included because not captured)	

In either case, the probabilities sum to 1.0. Each forms a multinomial likelihood that can be solved for p_1 and p_2. As noted earlier, the derived parameter N is then estimated as $M_{t+1}/[1-(1-\hat{p})(1-\hat{p})(1-\hat{p})]$ for data with no individual covariates.

Regardless of whether or not you include individuals not encountered in the likelihood, the key to understanding the fitting of closed capture models is in realizing that the event histories are governed entirely by the encounter probability. In fact, the process of estimating abundance for closed models is in effect the process of estimating detection probabilities – the probability that an animal will be caught for the first time (if at all), and the probability that if caught at least once, that it will be caught again. The different closed population models differ conceptually on how variation in the encounter probability (e.g., over time, among individuals) is handled. The mechanics of fitting these models in **MARK** is the subject of the rest of this chapter.

begin sidebar

What does 'closure' really mean?

The closed captures data types, as the name implies, all assume the population of interest is closed during the sampling period. Strictly speaking, the models assume that no births or deaths occur and no immigration or emigration occurs. Typically, we refer to a closed population as one that is free of unknown changes in abundance, as we can usually account for known changes. White *et al*. (1982: 3-4) provide a good overview of the closure assumption.

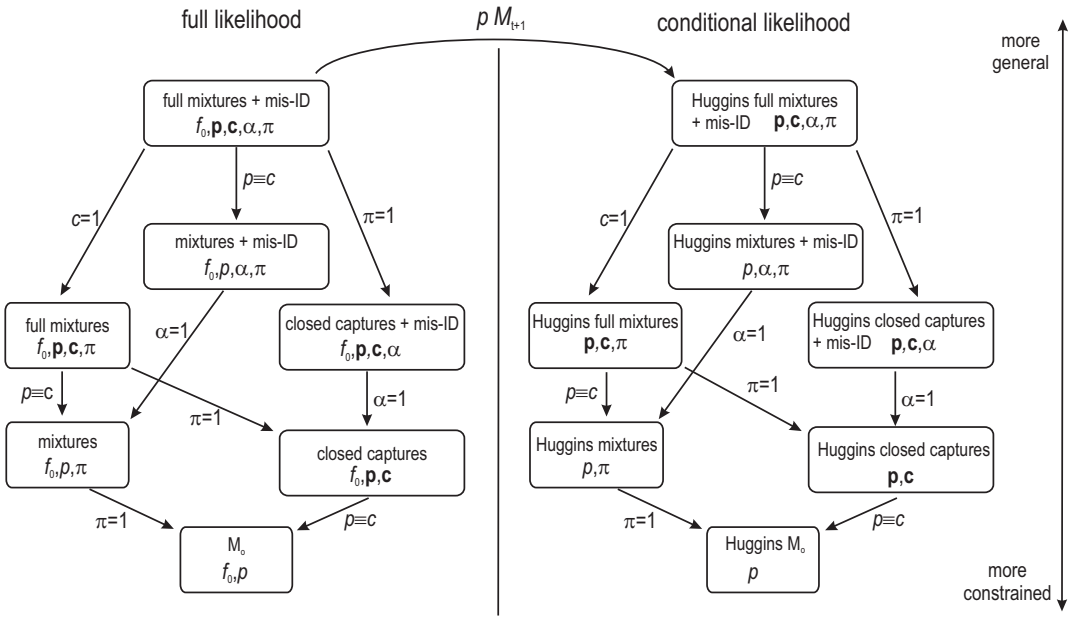
A few methods have been developed to test for closure violations. Program **CloseTest** exists to test the assumption of closure (Stanley and Burnham 1999). The Pradel model with recruitment parameterization has also been used to explore closure violations (Boulanger *et al*. 2002; see chapter 12 for details of the Pradel model). By analyzing closed population capture-recapture data with the Pradel recruitment parameterization, one could test for emigration and immigration. To test for emigration, compare a model with $\varphi = 1$ to a model with φ unconstrained. To test for immigration, compare a model with $f = 0$ to an model with f unconstrained. A likelihood ratio test could be used for the comparison.

Heterogeneity in capture probability can cloud our ability to detect closure violations. In situations where the population is truly closed, heterogeneity in capture probability can cause both the tests of immigration and emigration to reject the null hypothesis of closure.

end sidebar

14.3. Model types

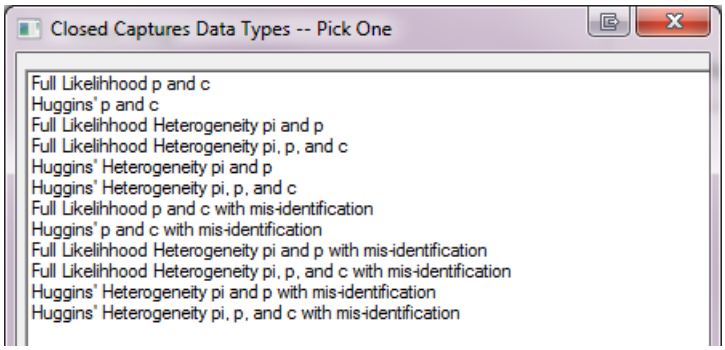
MARK currently supports 12 different closed population capture-recapture data types. These different data types can be classified within a hierarchy of dichotomous divisions – as shown in the diagram, below:



The first and most important split is between the models with abundance (or, rather, f_0) *in* the likelihood (Otis *et al*. 1978) and those with abundance *conditioned out* of the likelihood (Huggins 1989). We refer to the former as 'full likelihood' models, and the latter as either 'conditional likelihood' or 'Huggins' models. This is a major division that results in the two types of models not being comparable with standard AIC-based model selection techniques.

The remainder of the splits reflect one or more constraints on different parameters, and which parameters are included in the likelihoods. As noted earlier, the encounter histories in **MARK** are determined by the underlying encounter probabilities only. Minimally, most models in **MARK** are parameterized in terms of two different encounter parameters: p, the probability of first capture (i.e., the probability that an animal in the population will be captured - and marked - *for the very first time*), and c, the probability of recapture (conditional on having been captured at least once before). The c parameter is generally used to model for behavioral effects following inial capture. Both p and c can be time specific, although some specific constraints are required to ensure identifiability (discussed later). As a matter of convention in this chapter, we will use bold **p**'s and **c**'s to indicate a set (vector) of parameters that are (potentially) time varying, italic, un-subscripted p's and c's to indicate constant parameters, and italic, subscripted p's and c's refer to specific sampling occasions.

It is perhaps easiest to introduce the various models and parameters indicated in the preceding figure, by associating them with the different data types available in **MARK**. When you select '**closed captures**' in the data type specification window, **MARK** presents you with a popup window allowing you to select among these 12 different data types:



The first data type is labeled '**Full Likelihood p and c**'. These are the models of Otis *et al*. (1978). They are based on the full likelihood parametrization with three types of parameters; p_i, c_i, and f_0 (the number of individuals in the population, but not encountered).

The second data type is labeled '**Huggins p and c**'. These are the models of Huggins (1989). In this model, the likelihood is conditioned on the number of animals detected and f_0 therefore drops out of the likelihood. These models contain only p_i and c_i; the abundance N is estimated as a derived parameter. As noted earlier, the primary advantage of the Huggins data type is that individual covariates can be used to model p and c.

The next 4 model types are *heterogeneity* models. These models incorporate a *finite mixture* as an approximation to individual heterogeneity in the p_i parameter. In this model,

$$p_i = \begin{cases} p_{i,A} & \text{with } \Pr(\pi) \\ p_{i,B} & \text{with } \Pr(1-\pi) \end{cases}$$

for the case with two mixtures A and B, although the model can be extended to >2 mixtures. As written (above), the parameter π is the probability that the individual occurs in mixture A. For >2 mixtures, additional π parameters must be defined (i.e., π_A, π_B,...), but constrained to sum to 1.

Note that the '**heterogeneity models**' for both full likelihood closed captures and the Huggins' models come in one of two forms, differentiated by the presence of either (i) the mixture parameter, π, and both the p_i and c_i parameters, or (ii) the mixture parameter, π, and a single encounter parameter, p, only. The latter parameterizations (with only the π and p parameters) represent simple individual heterogeneity models, with parameters π, $p_{i,A} = p_A$, and $p_{i,B} = p_B$, and assume no temporal or behavioral variation. In contrast, the *full* parametrization models (including π, p and c parameters) provide for all three effects of time, behavior, and heterogeneity. Of course, any of the reduced models can be run from the full parameterizations if the appropriate constraints are applied.

The final six data types generalize the previous six data types to handle uncertainty in identification of individuals, typically from genotyping error (Lukacs & Burnham 2005). These models include an additional parameter, α, that is the probability that the individual was correctly identified on its first observation. In these models, N is estimated as a derived parameter. While it is possible to construct models for every data type using only the '**Full Likelihood heterogeneity pi, p, and c with**

mis-identification' and '**Huggins heterogeneity pi, p, and c with mis-identification**', the other data types are included to allow the user a less cumbersome set of parameters for building more constrained models.

Both the heterogeneity and misidentification models will be treated in more detail later in this chapter.

14.3.1. Constraining the final *p*

A subtlety of the closed population models is that the last p parameter is not identifiable unless a constraint is imposed. When no constraint is imposed on the last p_i , the likelihood is maximized with the last $p = 1$, giving the estimate $\hat{N} = M_{t+1}$. Why?

Consider a simple 2 occasion study. For this study, there are 4 possible encounter histories: `11`, `10`, `01`, and `00`. The expected frequencies for each history are:

history	*probability*
`11`	$p_1 c_2$
`10`	$p_1(1-c_2)$
`01`	$(1-p_1)p_2$
`00`	$(1-p_1)(1-p_2)$

Our interest concerns the final p parameter (in this case, p_2). We see that p_2 is a term in the probability expression for the '`01`' and '`00`' histories only. Taking the ratio of the *observed* frequency of '`00`' individuals to the *observed* frequency of '`01`' individuals (which is an *ad hoc* way of estimating p_2; see Chapter 1), then

$$\frac{f_{\{00\}}}{f_{\{01\}}} = \frac{(1-p_1)(1-p_2)}{(1-p_1)p_2} = \frac{(1-p_2)}{p_2}$$

Focus on the LHS of this expression. The numerator $f_{\{00\}}$ must be 0. Why? This must be true since the '`00`' history refers to individuals not seen. So, the observed frequency of animals not seen ($f_{\{00\}}$) is 0 (obviously), and thus the LHS of our equation is 0. We then solve for p_2 as

$$\begin{aligned} \frac{f_{\{00\}}}{f_{\{01\}}} &= \frac{(1-p_2)}{p_2} \\ 0 &= \frac{(1-p_2)}{p_2} \\ &= 1 - p_2 \end{aligned}$$

$$\therefore \quad \hat{p}_2 = 1$$

So, the final encounter probability p_2 is estimated at 1. OK - fine. But, why is that a problem? Recall that the canonical estimator for $\hat{N}$ is the count statistic (in this case, M_{t+1}) divided by the encounter probability. For a two occasion study,

$$\hat{N} = \frac{M_{t+1}}{\left(1 - [(1-\hat{p}_1)(1-\hat{p}_2)]\right)}$$

If $\hat{p}_2 = 1$, then

$$\begin{aligned}\hat{N} &= \frac{M_{t+1}}{\left(1 - [(1-\hat{p}_1)(1-\hat{p}_2)]\right)} \\ &= \frac{M_{t+1}}{\left(1 - [(1-\hat{p}_1)(1-1)]\right)} \\ &= \frac{M_{t+1}}{(1-0)} \\ &= M_{t+1}\end{aligned}$$

Thus, unless a constraint is placed on the last p, then the estimated abundance N will simply be M_{t+1}. Thus, it is diagnostic is to check to see whether $\hat{N} = M_{t+1}$, and if so, to see if the last p_i estimate equals 1. If they are, then you've forgotten to constrain p.

So, in model M_t, the constraint of $p_i = c_i$ is imposed, providing an estimate of the last p from the last c. Likewise, under model M_b, the constraint of $p_i = p_.$ is imposed, so that the last p is assumed equal to all the other p values. A similar constraint is used for model M_{bh}, i.e., $p_{i,A} = p_A, p_{i,B} = p_B$, and so on. Under model M_{tb}, the p_i and c_i are modeled as a constant offset (O_{beh}) of one another, i.e., $c_i = (p_i + O_{beh})$. This relationship will depend on the link function used, but the last p_i is still obtained as c_i minus the offset (where the offset is estimated from the data on the other p_i and c_i). Under model M_{tbh}, the offset between the p_i and c_i is applied, with an additional offset(s) included to model the relationship among the mixtures, i.e., $p_{i,B} = (p_{i,A} + O_B)$, $p_{i,C} = (p_{i,A} + O_C)$, with a different offset applied to each succeeding mixture. Similarly, $c_{i,B} = (p_{i,B} + O_{beh}) = (p_{i,A} + O_B + O_{beh})$, with the resulting relationship depending on the link function applied. With this model, the relationship between the mixtures of the p_i is maintained, i.e., the ordering of the mixtures is maintained across occasions. Model M_{th} can also be modeled as an additive offset between the mixtures, although other relationships are possible because the last p_i for each mixture is estimated from the corresponding last c_i. Although other relationships than those of the preceding paragraph can be proposed to provide identifiability, the proposed models must provide identifiability of all the initial capture probabilities.

14.4. Encounter histories format

All of the closed capture-recapture models use the LLLL encounter histories format (see chapter 2 for more detail). By the definition of a closed population, animals are not dying, therefore a dead encounter is not possible. On the same line of reasoning, time between sampling occasions is not relevant because there is no survival or movement process to consider. Encounter histories are followed by group frequencies. For the Huggins models, group frequencies can be followed with individual covariates. All encounter histories end with the standard semicolon.

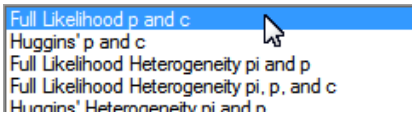
```
/* Closed capture-recapture data for a Huggins model.
   tag #, encounter history, males, females, length */
 /* 001 */   1001    1   0   22.3;
 /* 002 */   0111    1   0   18.9;
 /* 003 */   0100    0   1   20.6;
```

If you wish to analyze a data set that contains covariates in the input with both full and conditional likelihoods, you must initially import that data set by selecting a '**Huggins**' data type. The '**Closed**

'**Captures**' data type will not allow individual covariates to be specified. In this case, it is likely best to create two separate **MARK** files for the analysis because the AIC_c values are not comparable between the '**Closed Captures**' and '**Huggins**' data types.

14.5. Building models

Now it is time to move on to the actual mechanics of closed population abundance estimation in **MARK**. We will analyze some simulated data contained in (`simple_closed1.inp`). In this simulated data set (which consists of 6 encounter occasions), true $N = 350$. The total number of individuals encountered was $M_{t+1} = 339$ (so, 11 individuals were never seen). Open **MARK** and create a new database using the '**File | New**' option. Select the '**Closed Captures**' radio-button. When you click on the '**Closed Captures**' radio-button, a window will open that allows you to select a model type, shown earlier in this chapter. To start, select '**Full Likelihood p and c**'.



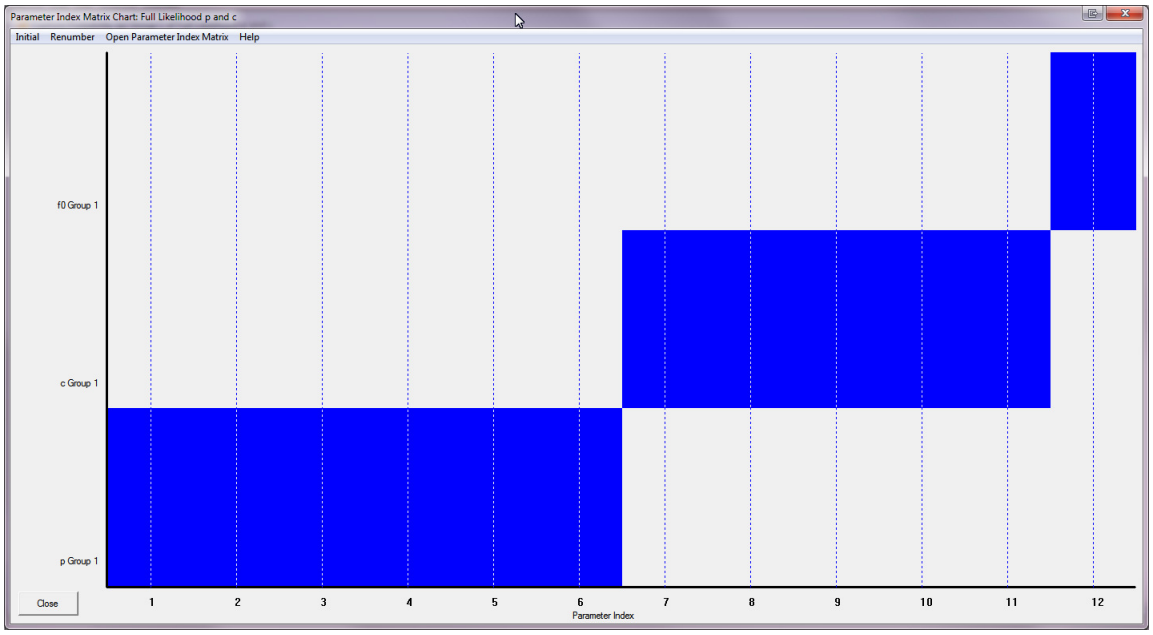
Enter a title, select the input file, and set the number of encounter occasions to 6.

To start, we'll construct some of the 'standard' closed capture models, as originally described in Otis *et al*. (1978). Model notation for the closed capture-recapture models in the literature often still follows that of Otis *et al*. (1978). Now that more complex models can be built, it seems appropriate to use a notation that is similar to the notation used for other models in **MARK**. Thus, our notation in this chapter will be based on a description of the parameters in the models. Below, we present a table contrasting model notation based on Otis *et al*. (1978) and expanded notation based on a description of the parameters. Combinations of the models described are possible.

Otis notation	*Expanded notation*	*Description*
M_0	$\{f_0, p(.) = c(.)\}$	Constant p
M_t	$\{f_0, p(t) = c(t)\}$	Time varying p
M_b	$\{f_0, p(.), c(.)\}$	Behavioral response
M_h or M_{h2}	$\{f_0, p_a(.) = c_a(.), p_b(.) = c_b(.), \pi\}$	Heterogeneous p

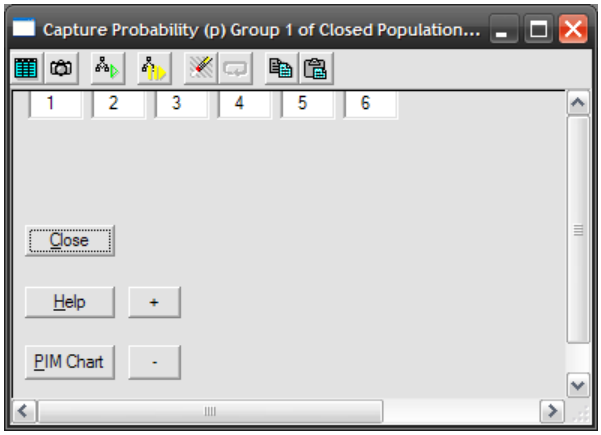
If you look closely at the 'expanded notation', you'll see that models are differentiated based on relationships between the p and c parameters. This is important - the closed capture-recapture models are one of the model types in **MARK** where different types of parameters are modeled as functions of each other. In this case p and c are commonly modeled as functions of one another. This makes intuitive sense because both p and c relate to catching animals.

With that said, let's begin building a few models to learn some of the tricks of using **MARK** to estimate abundance. We'll start with models $\{f_0, p(.) = c(.)\}$, $\{f_0, p(t) = c(t)\}$, and $\{f_0, p(.), c(.)\}$ (i.e., models M_0, M_t and M_b). Let's first examine the default PIM chart for the '**Full Likelihood p and c**' models (top of the next page).



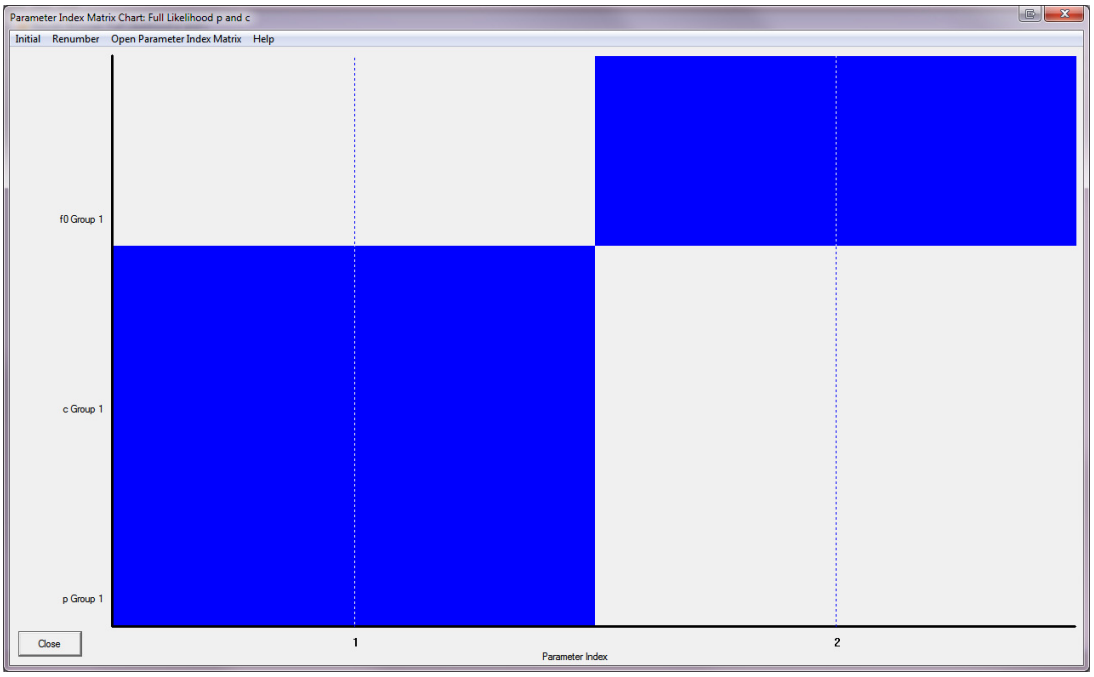
MARK defaults to a general time-varying parameter structure where there is a different p and c for each occasion. You may recall (from preceding section 14.3.1) that abundance is not estimable with this model structure because no constraint is imposed to estimate p_{10}. If this default, fully time-dependent model is fit to the data, $\hat{N} = M_{t+1}$ and $\hat{p}_{10} = 1.0$ regardless of the data. Therefore, in every model we build we must put some constraint on p_i for the last encounter occasion so that this parameter is estimated.

If we open the PIM windows (say, for p), we'll notice that the p's and c's have only a single row of text boxes.



In the closed capture models, every individual is assumed to be in the population and at risk of capture on every occasion. Therefore, there is no need for cohorts (expressed as multiple rows in the PIM window) as there is for many of the open-population models.

We'll start with $\{f_0, p(.) = c(.)\}$ - for this model, there is no temporal variation in either p or c, and the two parameters are set equal to each other. This model is easily constructed using the PIM chart:

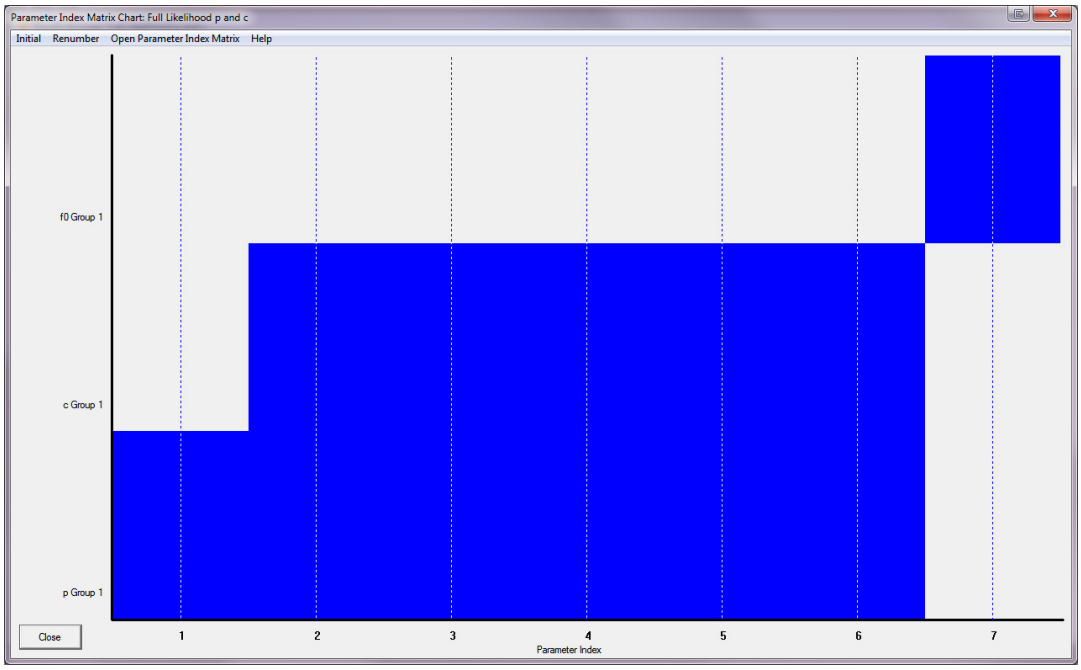


Go ahead and run this model, and add the results to the browser. Couple of important things to note. First, it is common for AIC_c values to be negative for the full likelihood closed captures models. Negative AIC_c values are legitimate and interpreted in the same way as positive AIC_c values. The negative number merely arises due to the portion of the multinomial coefficient that is computed. Keep in mind that minimum AIC_c remains the target and the model with the 'most negative' AIC_c, i.e., the one furthest from zero, is the most parsimonious model. AIC_c values from the conditional likelihood models are typically positive.

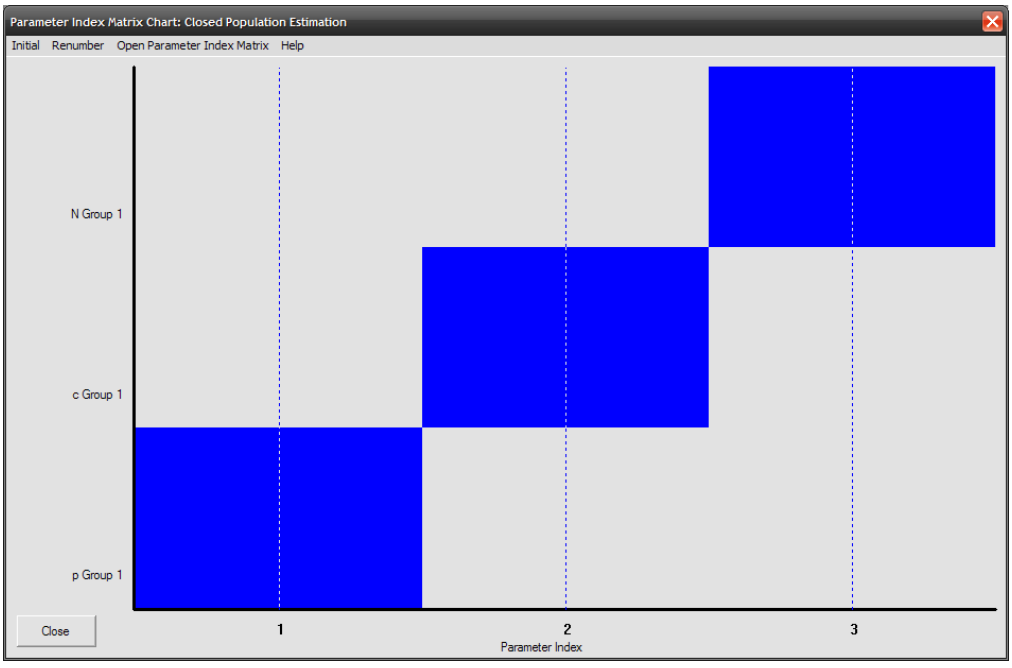
In addition, note that **MARK** defaults to a sin link, just as it does with all other data types when an identity design matrix is specified. In the case of the closed models, the sin link is used for the p's and c's, but a log link is used for f_0. The log link is used because f_0 must be allowed to be in the range of $[0 \rightarrow \infty]$. Therefore, no matter what link function you select, a log link will be used on f_0. If you choose the '**Parm-Specific**' option to set different link functions for each parameter, be sure you choose a link that does not constrain f_0 to the $[0, 1]$ interval. Choose either a log or identity link (log is preferable).

Now, we'll build model $\{N, p(t) = c(t)\}$ (i.e., model M_t). It is important to note that there is no c for the first occasion because it is impossible for an animal to be recaptured until it has been captured once. Therefore, **MARK** offers an easy way to assure that the correct p's line up with the correct c's: under the '**Initial**' menu select '**make c=p**' and renumber with overlap. The constraint on p_5 in this model is that $p_5 = c_5$.

Here is the PIM chart:



Finally, we'll build model $\{f_0, p(.), c(.)\}$ (i.e., model M_b). Here, we're accounting for possible differences in 'behavior' (i.e., encounter probability) between the first encounter, and subsequent encounters. Such a 'behavioral' effect might indicate some permanent 'trap effect' (trap 'happiness' or trap 'aversion'). For model $\{f_0, p(.), c(.)\}$, there is a 'behavior' effect, but no temporal variation. The PIM chart for this model is shown below:



Note that there is no 'overlap' (i.e., no function relating p and c) for this model – this is analogous to the default model $\{f_0, p(t), c(t)\}$, shown earlier. However, in this instance, all parameters are estimable because of the constraint that p and c are constant over time – the lack of estimability for the final p occurs if p is time dependent. As such, model $\{f_0, p(.), c(t)\}$ would be estimable, while model $\{f_0, p(t), c(.)\}$ would not (for this model $\hat{N} = M_{t+1}$). You might want to confirm this for yourself.

begin sidebar

simple extension - removal models

Now let's consider a *removal* model. These are commonly used in fisheries work where the researcher does not want to subject a fish to multiple passes of electricity. Therefore, the fish that are encountered are held aside until all sampling has occurred.

To accomplish this in **MARK**, build an $\{f_0, p(t), c(.)\}$ or $\{f_0, p(.), c(.)\}$ model. Then click '**Run**' to open the run window. Click the '**fix parameters**' button. A window will open listing all of the real parameters in the model. Simply fix $c = 0$, and run the model. Note - a removal model requires that the number of captures decrease with occasion.

end sidebar

14.6. Closed population models and the design matrix

In the preceding, we constructed 3 simple models using the PIM chart. While using the PIM chart was very straightforward for those models, through the design matrix **MARK** allows models to be fit that were not possible with the PIM chart. For example, it is possible to build an $\{f_0, p(t) = c(t) + b\}$ model where capture probability and recapture probability are allowed to vary through time, but constrained to be different by an additive constant on the logit scale. It is also worth noting that these extended models are not available in program **CAPTURE** (representing one of several reasons that **CAPTURE** is no longer preferred for fitting closed population abundance models).

As introduced in Chapter 6, one approach to doing this is to first build a general model using PIMs, and then construct the design matrix corresponding to this general model. Then, once you have the general model constructed using the design matrix, all other models of interest can be constructed simply by modifying the design matrix. In this case, the most general model we can build is $\{f_0, p(t), c(t)\}$. As noted above, we know before the fact that this particular model is not a useful model, but it is convenient to build the design matrix for this model as a starting point.

To do this we need the PIMs in the full time varying setup (again, as shown at the top of p. 11). Go ahead and run this model, and add the results to the browser. Look at the real and derived parameter estimates – note that (i) $\hat{p}_5 = 1.0$, and (ii) $\hat{N} = M_{t+1} = 339$. Note as well that the reported SEs for both $\hat{p}_5$ and $\hat{N}$ are impossibly small – a general diagnostic that there is 'something wrong' with this model (as expected). As discussed earlier, this is not a useful model without imposing some constraints since the estimate of $\hat{N} = M_{t+1}$.

Now, the design matrix. Recall that there are 12 parameters specifying this model: $1 \rightarrow 6$ for p, $7 \rightarrow 11$ for c, and parameter 12 for abundance, N. Thus, our design matrix will have 12 columns. Now, if you select '**Design | Full**', **MARK** will respond with the default DM shown at the top of the next page:

B1: p Int	B2: p t1	B3: p t2	B4: p t3	B5: p t4	B6: p t5	Parm	B7: c Int	B8: c t1	B9: c t2	B10: c t3	B11: c t4	B12: f0 Int
1	1	0	0	0	0	1:p	0	0	0	0	0	0
1	0	1	0	0	0	2:p	0	0	0	0	0	0
1	0	0	1	0	0	3:p	0	0	0	0	0	0
1	0	0	0	1	0	4:p	0	0	0	0	0	0
1	0	0	0	0	1	5:p	0	0	0	0	0	0
1	0	0	0	0	0	6:p	0	0	0	0	0	0
0	0	0	0	0	0	7:c	1	1	0	0	0	0
0	0	0	0	0	0	8:c	1	0	1	0	0	0
0	0	0	0	0	0	9:c	1	0	0	1	0	0
0	0	0	0	0	0	10:c	1	0	0	0	1	0
0	0	0	0	0	0	11:c	1	0	0	0	0	0
0	0	0	0	0	0	12:f0	0	0	0	0	0	1

Here, we see a DM which is strictly analogous to what we might have expected for 3 parameters - each parameter (in this case, p, c and f_0) has a separate 'block' within the matrix: p in the upper-left, c in the middle, and f_0 in the lower-right. If you go ahead and run this model, you'll see (below) that it gives you exactly the same model deviance as the general model built with PIMs.

Model	AICc	Delta AICc	AICc Weight	Model Likelihood	No. Par.	Deviance
{f0,p(t),c(t) - DM)}	-530.1030	0.0000	0.53930	1.0000	10	41.6249
{f0,p(t)=c(t)}	-528.6779	1.4251	0.26447	0.4904	7	49.1035
{f0,p(t),c(t) - PIM}	-528.0811	2.0219	0.19624	0.3639	11	41.6249
{f0,p(.),c(.)}	-501.8661	28.2369	0.00000	0.0000	3	83.9587
{f0,p(.)=c(.)}	-494.4003	35.7027	0.00000	0.0000	2	93.4304

You'll also note, however, that the AIC_c reported for this DM-built general model is not the same as the AIC_c reported for the general model built with PIMs (-530.1030 versus -528.0812). If the model deviances are the same, but the reported AIC_c values are different, then this implies that the number of estimated parameters is different. In fact, we see that the number estimated for the 'full default DM' model is 10, whereas for the model built with PIMs, the number reported is 11. In fact, for this model, the difference in the number reported isn't particularly important, since this is not a 'reasonable' model in the first instance (as mentioned several times earlier in this chapter). The fact that the model deviances 'match' indicates that the DM is correct.

However, while this is 'technically' true, the default DM assumes that there is no interest in creating a functional relationship between any of the parameters. While normally this is a reasonable assumption (e.g., in a CJS live encounter study, there is no plausible reason to create a functional relationship between φ and p), this is clearly not the case for closed population abundance models, where many of the models of interest are specified by imposing a particular relationship between p and c. For example, model $\{f_0, p(t) = c(t)\}$ imposes a relationship between p and c at each sampling occasion t.

How do we accommodate our interest in specifying these relationships between p and c in the DM? In fact, it is very easy, with a simple conceptual 'trick' - we're going to treat the two parameters p and c as if they were levels of some putative 'treatment' - in precisely the same way (structurally) that we handled age (TSM) effects for individuals marked as young in age (TSM) models (Chapter 7 – section 7.2). As a reminder, recall how we would construct the design matrix to correspond to the PIM for survival for a simple age model, with 2 age classes, and time-dependence in each age class. Assume that we have 7 occasions.

Recall that the PIM for this model looks like:

```
 1   7   8   9  10  11
     2   8   9  10  11
         3   9  10  11
             4  10  11
                5  11
                   6
```

So, based on the number of indexed parameters in the PIM, we know already that our design matrix for survival would need to have 11 rows and 11 columns.

What does the linear model look like? Again, writing out the linear model is often the easiest place to start. In this case we see that over a given time interval, we have, in effect, 2 kinds of individuals: *juveniles* (individuals in their first year after marking), and *adults* (individuals at least 2 years after marking). Thus, for a given `TIME` interval, there are 2 groups: juvenile and adult. If we call this group effect `AGE`, then we can write out our linear model as

$$\begin{aligned}\text{'survival'} &= \texttt{AGE+TIME+AGE.TIME} \\ &= \beta_1 \\ &\quad + \beta_2(\texttt{AGE}) \\ &\quad + \beta_3(\texttt{T}_1) + \beta_4(\texttt{T}_2) + \beta_5(\texttt{T}_3) + \beta_6(\texttt{T}_4) + \beta_7(\texttt{T}_5) \\ &\quad + \beta_8(\texttt{AGE.T}_2) + \beta_9(\texttt{AGE.T}_3) + \beta_{10}(\texttt{AGE.T}_4) + \beta_{11}(\texttt{AGE.T}_5)\end{aligned}$$

Again, recall from Chapter 7 that there is no ($\texttt{AGE.T}_1$) interaction term. Also remember, we're treating the two age classes as different groups - this will be the key 'conceptual step' in seeing how we apply the same idea to closed population abundance models.

The design matrix corresponding to this linear model is:

Design Matrix Specification: Live Recaptures (CJS)

Design Matrix Specification (B = Beta)

B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	B11	Parm
1	1	1	0	0	0	0	0	0	0	0	1:Phi
1	1	0	1	0	0	0	1	0	0	0	2:Phi
1	1	0	0	1	0	0	0	1	0	0	3:Phi
1	1	0	0	0	1	0	0	0	1	0	4:Phi
1	1	0	0	0	0	1	0	0	0	1	5:Phi
1	1	0	0	0	0	0	0	0	0	0	6:Phi
1	0	0	1	0	0	0	0	0	0	0	7:Phi
1	0	0	0	1	0	0	0	0	0	0	8:Phi
1	0	0	0	0	1	0	0	0	0	0	9:Phi
1	0	0	0	0	0	1	0	0	0	0	10:Phi
1	0	0	0	0	0	0	0	0	0	0	11:Phi

So, column `B2` in this design matrix indicates a putative 'age group' - for a given cohort, and a given time step, is the individual young (indicated with the dummy '1') or adult (indicated with the dummy '0'). If you don't recall this connection, go back and re-read section 7.2.

Now, what does this have to do with building design matrices for closed abundance estimation models? The connection relates to the idea of creating a 'logical group'. For age models, we used the age of an individual for a given cohort and time step as a grouping variable. For closed population abundance models, we do the same thing - except that instead of age, we're going to 'group' as a function of whether or not the individual has been captured at least once or not. In other words, we're going to treat the parameters p (caught for the first time) and c (caught subsequently) as levels of a putative 'encounter' group (analogous to young and adult, respectively).

This will make more sense when you see how we set up the DM. Here it is - note that it is *identical* to the age (TSM) model (above):

B1 intercept	B2 enc grp	B3 t1	B4 t2	B5 t3	B6 t4	B7 t5	Parm	B8 eg*t2	B9 eg*t3	B10 eg*t4	B11 eg*t5	B12 f0
1	1	1	0	0	0	0	1:p	0	0	0	0	0
1	1	0	1	0	0	0	2:p	1	0	0	0	0
1	1	0	0	1	0	0	3:p	0	1	0	0	0
1	1	0	0	0	1	0	4:p	0	0	1	0	0
1	1	0	0	0	0	1	5:p	0	0	0	1	0
1	1	0	0	0	0	0	6:p	0	0	0	0	0
1	0	0	1	0	0	0	7:c	0	0	0	0	0
1	0	0	0	1	0	0	8:c	0	0	0	0	0
1	0	0	0	0	1	0	9:c	0	0	0	0	0
1	0	0	0	0	0	1	10:c	0	0	0	0	0
1	0	0	0	0	0	0	11:c	0	0	0	0	0
0	0	0	0	0	0	0	12:f0	0	0	0	0	1

Column `B1` is the common intercept - this is a necessary step (and a key difference from the default DM) in order to allow us to specify a functional relationship between p and c. Column `B2` is the column which specifics the putative 'encounter group' - first encounter (corresponding to parameter p) or subsequent encounter (corresponding to parameter c). Note that there are 6 '1's; for p, but only 5 '0's' for c (since there is no c parameter for occasion 1). This is *entirely analogous* to having no adults in the first occasion for individuals marked as young. Columns `B3` → `B7` correspond to the time steps - again, note that for parameter c, there is no time coding for interval 1. These are followed by the interaction columns `B8` → `B11`. Again, there is no logical interaction of p and c for occasion 1 (since there is no parameter c_1), so the interaction columns start with time interval 2. Finally, column `B12` for the parameter f_0.

Go ahead, run this model, and add the results to the browser:

Model	AICc	Delta AICc	AICc Weight	Model Likelihood	No. Par.	Deviance
{f0,p(t),c(t) - DM)}	-530.1030	0.0000	0.35035	1.0000	10	41.6249
{f0,p(t),c(t) - DM - common intercept}	-530.1030	0.0000	0.35035	1.0000	10	41.6249
{f0,p(t)=c(t)}	-528.6779	1.4251	0.17181	0.4904	7	49.1035
{f0,p(t),c(t) - PIM}	-528.0811	2.0219	0.12748	0.3639	11	41.6249
{f0,p(.),c(.)}	-501.8661	28.2369	0.00000	0.0000	3	83.9587
{f0,p(.)=c(.)}	-494.4003	35.7027	0.00000	0.0000	2	93.4304

We see that the model deviances for the general model constructed with (i) PIMs, (ii) the default DM (which used a separate intercept for each parameter), and (iii) the modified DM which used a common intercept, are all identical.

Now, let's see how to use the DM to build the 3 models we constructed previously using PIMs. First, model $\{f_0, p(.) = c(.)\}$. We see that (i) there is no temporal variation (meaning, we simply delete the

columns corresponding to time and interactions with time from the DM – columns B3 → B11), and (ii) $p = c$ (meaning, we delete the column specifying difference between the putative 'encounter groups' - column B2):

B1: intercept	Parm	B2: f0
1	1:p	0
1	2:p	0
1	3:p	0
1	4:p	0
1	5:p	0
1	6:p	0
1	7:c	0
1	8:c	0
1	9:c	0
1	10:c	0
1	11:c	0
0	12:f0	1

Run this model and add the results to the browser:

Model	AICc	Delta AICc	AICc Weight	Model Likelihood	No. Par.	Deviance
{f0,p(t),c(t) - DM)}	-530.1030	0.0000	0.35035	1.0000	10	41.6249
{f0,p(t),c(t) - DM - common intercept}	-530.1030	0.0000	0.35035	1.0000	10	41.6249
{f0,p(t)=c(t)}	-528.6779	1.4251	0.17181	0.4904	7	49.1035
{f0,p(t),c(t) - PIM}	-528.0811	2.0219	0.12748	0.3639	11	41.6249
{f0,p(.),c(.)}	-501.8661	28.2369	0.00000	0.0000	3	83.9587
{f0,p(.)=c(.)}	-494.4003	35.7027	0.00000	0.0000	2	93.4304
{f0,p(.)=c(.) - DM coding}	-494.4003	35.7027	0.00000	0.0000	2	93.4304

We see the model results match those of the same model constructed using PIMs.

What about model $\{f_0, p(.), c(.)\}$? Here, we again delete all of the time and interaction columns, but retain the column coding for the 'encounter group' term in the model:

B1: intercept	Parm	B2: enc grp	B3: f0
1	1:p	1	0
1	2:p	1	0
1	3:p	1	0
1	4:p	1	0
1	5:p	1	0
1	6:p	1	0
1	7:c	0	0
1	8:c	0	0
1	9:c	0	0
1	10:c	0	0
1	11:c	0	0
0	12:f0	0	1

Again, we see that the results of fitting this model constructed using the DM approach exactly match those from the same model constructed using PIMs (as indicated in the result browser shown at the top of the next page).

Model	AICc	Delta AICc	AICc Weight	Model Likelihood	No. Par.	Deviance
{f0,p(t),c(t) - DM)}	-530.1030	0.0000	0.35035	1.0000	10	41.6249
{f0,p(t),c(t) - DM - common intercept}	-530.1030	0.0000	0.35035	1.0000	10	41.6249
{f0,p(t)=c(t)}	-528.6779	1.4251	0.17181	0.4904	7	49.1035
{f0,p(t),c(t) - PIM}	-528.0811	2.0219	0.12748	0.3639	11	41.6249
{f0,p(.),c(.)}	-501.8661	28.2369	0.00000	0.0000	3	83.9587
{f0,p(.),c(.) - DM coding}	-501.8661	28.2369	0.00000	0.0000	3	83.9587
{f0,p(.)=c(.)}	-494.4003	35.7027	0.00000	0.0000	2	93.4304
{f0,p(.)=c(.) - DM coding}	-494.4003	35.7027	0.00000	0.0000	2	93.4304

Finally, model $\{f_0, p(t) = c(t)\}$. Here, we have no 'encounter group' effect, but simple temporal variation in p and c. We simply delete the interaction and 'encounter group' columns:

B1: intercept	Parm	B2: t1	B3: t2	B4: t3	B5: t4	B6: t5	B7: f0
1	1:p	1	0	0	0	0	0
1	2:p	0	1	0	0	0	0
1	3:p	0	0	1	0	0	0
1	4:p	0	0	0	1	0	0
1	5:p	0	0	0	0	1	0
1	6:p	0	0	0	0	0	0
1	7:c	0	1	0	0	0	0
1	8:c	0	0	1	0	0	0
1	9:c	0	0	0	1	0	0
1	10:c	0	0	0	0	1	0
1	11:c	0	0	0	0	0	0
0	12:f0	0	0	0	0	0	1

We see (below) that the model deviances are identical, regardless of whether or not the PIM or DM approach was used.

Model	AICc	Delta AICc	AICc Weight	Model Likelihood	No. Par.	Deviance
{f0,p(t),c(t) - DM)}	-530.1030	0.0000	0.29898	1.0000	10	41.6249
{f0,p(t),c(t) - DM - common intercept}	-530.1030	0.0000	0.29898	1.0000	10	41.6249
{f0,p(t)=c(t)}	-528.6779	1.4251	0.14662	0.4904	7	49.1035
{f0,p(t)=c(t) - DM}	-528.6779	1.4251	0.14662	0.4904	7	49.1035
{f0,p(t),c(t) - PIM}	-528.0811	2.0219	0.10879	0.3639	11	41.6249
{f0,p(.),c(.)}	-501.8661	28.2369	0.00000	0.0000	3	83.9587
{f0,p(.),c(.) - DM coding}	-501.8661	28.2369	0.00000	0.0000	3	83.9587
{f0,p(.)=c(.)}	-494.4003	35.7027	0.00000	0.0000	2	93.4304
{f0,p(.)=c(.) - DM coding}	-494.4003	35.7027	0.00000	0.0000	2	93.4304

Now, let's consider a model which we couldn't build using the PIM-only approach (or, as noted, if we'd relied on the default DM) – a model with an additive 'offset' between p and c. As we introduced in Chapter 6, to build such additive models, all you need to do is delete the interaction columns from the DM – this 'additive' model is shown at the top of the next page. Remember that this model constrains time-specific estimates of p and c to parallel each other by a constant offset. In effect, this is a 'behavior+time' model. Whether or not this is a 'meaningful' model is up to you.

B1: intercept	B2: enc grp	Parm	B3: t1	B4: t2	B5: t3	B6: t4	B7: t5	B8: f0
1	1	1:p	1	0	0	0	0	0
1	1	2:p	0	1	0	0	0	0
1	1	3:p	0	0	1	0	0	0
1	1	4:p	0	0	0	1	0	0
1	1	5:p	0	0	0	0	1	0
1	1	6:p	0	0	0	0	0	0
1	0	7:c	0	1	0	0	0	0
1	0	8:c	0	0	1	0	0	0
1	0	9:c	0	0	0	1	0	0
1	0	10:c	0	0	0	0	1	0
1	0	11:c	0	0	0	0	0	0
0	0	12:f0	0	0	0	0	0	1

14.7. Heterogeneity models

As noted earlier, **MARK** allows you to fit a class of models which are parameterized based on what are known as 'finite mixtures'.* These models have proven to be very useful for modeling unspecified heterogeneity among individuals in the p_i and c_i parameters. In these models,

$$p_i = \begin{cases} p_{i,A} & \text{with } \Pr(\pi) \\ p_{i,B} & \text{with } \Pr(1-\pi) \end{cases}$$

for the case with two mixtures A and B, although the model can be extended to >2 mixtures. As written (above), the parameter π is the probability that the individual occurs in mixture A. For >2 mixtures, additional π parameters must be defined (i.e., π_A, π_B,...), but constrained to sum to 1.† In practice, most data sets generally support no more than 2 mixtures. Note that the π parameter is assumed to be constant over time (i.e., an individual in a given mixture is always in that particular mixture over the sampling period). This has important implications for constructing the DM, which we discuss later.

Before we demonstrate the 'mechanics' of fitting finite mixture models to the data, let's first consider the encounter histories and their probabilities for a 4-occasion case for the '**`Full likelihood p and c`**' data type:

history	*cell probability*
1000	$p_1(1-c_2)(1-c_3)(1-c_4)$
0100	$(1-p_1)p_2(1-c_3)(1-c_4)$
0010	$(1-p_1)(1-p_2)p_3(1-c_4)$
0001	$(1-p_1)(1-p_2)(1-p_3)p_4$
1100	$p_1c_2(1-c_3)(1-c_4)$
1010	$p_1(1-c_2)c_3(1-c_4)$
1001	$p_1(1-c_2)(1-c_3)c_4$
1110	$p_1c_2c_3(1-c_4)$

history	*cell probability*
1101	$p_1c_2(1-c_3)c_4$
1011	$p_1(1-c_2)c_3c_4$
0110	$(1-p_1)p_2c_3(1-c_4)$
0101	$(1-p_1)p_2(1-c_3)c_4$
0011	$(1-p_1)(1-p_2)p_3c_4$
0111	$(1-p_1)p_2c_3c_4$
1111	$p_1c_2c_3c_4$
0000	$(1-p_1)(1-p_2)(1-p_3)(1-p_4)$

* Here, we introduce the application of 'finite mixture models' to closed population abundance estimation. In fact, 'finite mixture models' are available for a number of additional data types in **MARK** – see the Addendum to this chapter

† In practice, this means that you should use the multinomial logit link function, MLogit, to ensure that the estimates do sum to 1. The MLogit link was introduced in Chapter 10.

There are 2^k possible encounter histories for a k-occasion study.

If we want to add a *finite mixture* to the cell probability (i.e., for '**Full Likelihood Heterogeneity with pi, p, and c**' data type, with two mixtures), we modify the probability expressions as follows:

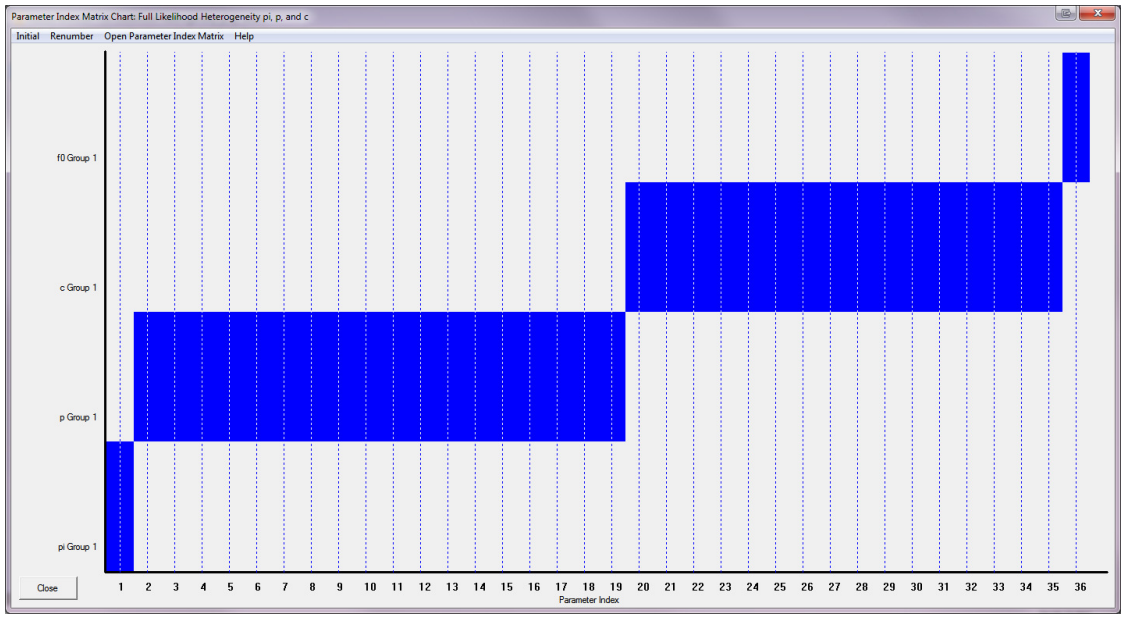
history	*cell probability*
1000	$\sum_{a=1}^{2}(\pi_a p_{a1}(1-c_{a2})(1-c_{a3})(1-c_{a4}))$
0100	$\sum_{a=1}^{2}(\pi_a (1-p_{a1})p_{a2}(1-c_{a3})(1-c_{a4}))$
0010	$\sum_{a=1}^{2}(\pi_a (1-p_{a1})(1-p_{a2})p_{a3}(1-c_{a4}))$
0001	$\sum_{a=1}^{2}(\pi_a (1-p_{a1})(1-p_{a2})(1-p_{a3})p_{a4})$
1100	$\sum_{a=1}^{2}(\pi_a p_{a1}c_{a2}(1-c_{a3})(1-c_{a4}))$
1010	$\sum_{a=1}^{2}(\pi_a p_{a1}(1-c_{a2})c_{a3}(1-c_{a4}))$
1001	$\sum_{a=1}^{2}(\pi_a p_{a1}(1-c_{a2})(1-c_{a3})c_{a4})$
1110	$\sum_{a=1}^{2}(\pi_a p_{a1}c_{a2}c_{a3}(1-c_{a4}))$
1101	$\sum_{a=1}^{2}(\pi_a p_{a1}c_{a2}(1-c_{a3})c_{a4})$
1011	$\sum_{a=1}^{2}(\pi_a p_{a1}(1-c_{a2})c_{a3}c_{a4})$
0110	$\sum_{a=1}^{2}(\pi_a (1-p_{a1})p_{a2}c_{a3}(1-c_{a4}))$
⋮	⋮

Note: the finite mixture models have a separate set of p's and c's for each mixture.

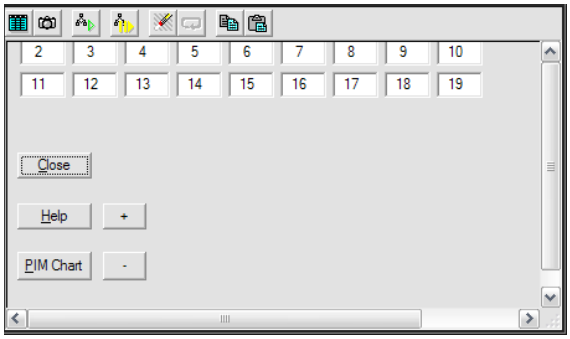
We will demonstrate the fitting of finite mixture ('heterogeneity') models to a new sample data set (`mixed_closed1.inp`). These data were simulated assuming a finite mixture (i.e., heterogeneity) using the generating model $\{N, \pi, p(.) = c(.) = constant\}$ - 9 occasions, 2 mixtures, N = 2,000, $\pi = 0.40$, and $p_{\pi_A} = 0.25, p_{\pi_B} = 0.75$. In other words, two mixtures, one with an encounter probability of $p = 0.25$, the other with an encounter probability of $p = 0.75$, with the probability of being in the first mixture $\pi = 0.40$.

Start a new project, select the input data file, set the number of occasions to 9, and specify the '**Full Likelihood Heterogeneity with pi, p, and c'** data type. Once we've selected a closed data type with heterogeneity, you'll see that an option to specify the number of mixtures is now available in the '**specification window**' (lower-right side). We'll use 2 mixtures for this example.

Once you have specified the number of mixtures, open the PIM chart for this data type (when you switch data types, the underlying model will default to a general time-specific model) – the PIM chart is shown at the top of the next page. You'll notice immediately that there are now twice as many p's and c's as you might have expected given there are 9 occasions represented in the data. This increase represents the parameters for each of the two mixture groups.



The PIM for the p's now has two rows defaulting to parameters $2 \rightarrow 10$ and $11 \rightarrow 19$.



Parameters $2 \rightarrow 10$ represent the p's for the first mixture and $11 \rightarrow 19$ represent the p's for the second mixture. It becomes more important with the mixture models to keep track of which occasion each c corresponds to because now *both* parameter 2 and 11 relate to occasion 1 which has no corresponding c parameter.

We'll follow the approach used in the preceding section, by first fitting a general model based on PIMs to the data. You might consider model $\{f_0, \pi, p(t), c(t)\}$ as a reasonable starting model. However, there are two problems with using this as a general, starting model. First, you'll recall that there are estimation problems (in general) for a closed abundance model where both p and c are fully time-dependent. Normally, we need to impose some sort of constraint to achieve identifiability. However, even if we do so, there is an additional, more subtle problem here – recall we are fitting a heterogeneity 'mixture' model, where the parameter π is assumed to be constant over time. As such, there is no interaction among mixture groups possible over time. Such an interaction would imply time-varying π. Thus, the most general model we could fit would be an additive model, with additivity between the mixture groups. We'll allow for possible interaction of p and c within a given mixture group. Recall that we can't construct this model using PIMs - building an additive model requires use of the design matrix.

We see from the PIM chart (shown on the preceding page) that the default reduced DM has 36 columns. Note: if you select '**`Design | Full`**', **MARK** will respond with an error message, telling you it can't build a default fully time-dependent DM. Basically, for heterogeneity models, you'll need to build the DM by hand - meaning, starting with a reduced DM. So, we select '**`Design | Reduced`**', and keep the default 36 columns (we can change the number of columns manually later, if need be).

Now, how do we build the DM? We might start by first writing out the linear model. To do so, we need to first consider the 'groups' in our model. Here, we have in fact 2 groups: (i) the putative 'encounter group' (`ENCGRP`) representing the p and c parameters (as we saw in the preceding section), and (ii) a new 'heterogeneity' group (`HETGRP`) representing what we might for convenience call the 'π' and '$1-\pi$' groups. So, two ' encounter groups', 2 'heterogeneity groups', 9 occasions (`TIME`), and the various *plausible* interactions among them.

Here is our linear model (which we write only in terms of parameters p and c. Parameters π and N are simple scalar constants):

$$
\begin{aligned}
f &= \texttt{ENCGRP+HETGRP+TIME+(ENCGRP.TIME)+(HETGRP.TIME)+(ENCGRP.HETGRP.TIME)} \\
&= \beta_1 \\
&\quad + \beta_2(\texttt{ENCGRP}) \\
&\quad + \beta_3(\texttt{HETGRP}) \\
&\quad + \beta_4(\texttt{T}_1) + \beta_5(\texttt{T}_2) + \beta_6(\texttt{T}_3) + \beta_7(\texttt{T}_4) + \beta_8(\texttt{T}_5) + \beta_9(\texttt{T}_6) + \beta_{10}(\texttt{T}_7) + \beta_{11}(\texttt{T}_8) \\
&\quad + \beta_{12}(\texttt{HETGRP.T}_1) + \beta_{13}(\texttt{HETGRP.T}_2) + \beta_{14}(\texttt{HETGRP.T}_3) + \cdots + \beta_{19}(\texttt{HETGRP.T}_8) \\
&\quad + \beta_{20}(\texttt{ENCGRP.T}_2) + \beta_{21}(\texttt{ENCGRP.T}_3) + \beta_{22}(\texttt{ENCGRP.T}_4) + \cdots + \beta_{26}(\texttt{ENCGRP.T}_8) \\
&\quad + \beta_{27}(\texttt{ENCGRP.HETGRP.T}_1) + \beta_{28}(\texttt{ENCGRP.HETGRP.T}_2) + \cdots + \beta_{34}(\texttt{ENCGRP.HETGRP.T}_8)
\end{aligned}
$$

So, 34 parameters in this linear model. If we add 2 (for π and N, respectively), we get 36 total. The design matrix corresponding to this model is shown below (you might need to put on some 'special reading glasses' to see it all):

B1 pi	B2 intercept	B3 H	B4 E	B5 t1	B6 t2	B7 t3	B8 t4	B9 t5	B10 t6	B11 t7	B12 t8	Parm	B13 H*t1	B14 H*t2	B15 H*t3	B16 H*t4	B17 H*t5	B18 H*t6	B19 H*t7	B20 H*t8	B21 E*t2	B22 E*t3	B23 E*t4	B24 E*t5	B25 E&T6	B26 E*T7	B27 E*T8	B28 H*E*t1	B29 H*E*t2	B30 H*E*t3	B31 H*E*t4	B32 H*E*t5	B33 H*E*t6	B34 H*E*t7	B35 H*E*t8	B36 f0
1	0	0	0	0	0	0	0	0	0	0	0	1pi	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	1	1	1	0	0	0	0	0	0	0	2p	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
0	1	1	1	0	1	0	0	0	0	0	0	3p	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
0	1	1	1	0	0	1	0	0	0	0	0	4p	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0
0	1	1	1	0	0	0	1	0	0	0	0	5p	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0
0	1	1	1	0	0	0	0	1	0	0	0	6p	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0
0	1	1	1	0	0	0	0	0	1	0	0	7p	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0
0	1	1	1	0	0	0	0	0	0	1	0	8p	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0
0	1	1	1	0	0	0	0	0	0	0	1	9p	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0
0	1	1	1	0	0	0	0	0	0	0	0	10p	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	1	0	1	0	0	0	0	0	0	0	11p	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	1	0	0	1	0	0	0	0	0	0	12p	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0	0	13p	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	1	0	0	0	0	1	0	0	0	0	14p	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	1	0	0	0	0	0	1	0	0	0	15p	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	1	0	0	0	0	0	0	1	0	0	16p	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
0	1	1	0	0	0	0	0	0	0	1	0	17p	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
0	1	1	0	0	0	0	0	0	0	0	1	18p	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
0	1	1	0	0	0	0	0	0	0	0	0	19p	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	1	0	0	0	0	0	0	0	0	20c	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	1	0	1	0	0	0	0	0	0	21c	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	1	0	0	1	0	0	0	0	0	22c	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	1	0	0	0	1	0	0	0	0	23c	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	1	0	0	0	0	1	0	0	0	24c	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	1	0	0	0	0	0	1	0	0	25c	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	1	0	0	0	0	0	0	1	0	26c	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	1	0	0	0	0	0	0	0	1	27c	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0	0	28c	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	1	0	0	0	0	0	29c	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	1	0	0	0	0	30c	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	1	0	0	0	31c	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	1	0	0	32c	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	1	0	33c	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	0	1	34c	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	0	0	35c	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	36f0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Now, some important things to note from the linear model and corresponding DM. First, the two 'groups' (encounter and heterogeneity; `ENCGRP` and `HETGRP`, respectively) are each coded by a single

column (single β) - columns `B3` and `B4`. 9 sampling occasions, so 8 columns for time (`B5` → `B12`). The remaining columns code for the two-way interactions between `ENCGRP (E)`, `HETGRP (H)` and time `(T)`, and the three-way interaction (`H.E.T`$_x$)

Now, if you run this model, and look at the parameter estimates, you'll quickly notice that some of the parameters aren't identifiable. In particular, the final $\hat{p}$ estimates for the two mixture groups have problems, and the estimate of $\hat{N}$ is simply M_{t+1} (the SE of the abundance estimate is clearly wrong).

Why the problems? Simple - despite the fact we have 2 mixture groups, this is still model $\{p(t), c(t)\}$, which we know is not identifiable without constraints. One possible constraint is to model p and c as additive functions of each other. How can we modify the DM to apply this constraint? Simply, by eliminating the interactions between `ENCGRP` and `TIME`. In other words, deleting columns `B21` → `B27` (coding for the interaction of `ENCGRP` and `TIME`), and columns `B28` → `B35` (coding for the 3-way interaction of `HETGRP`, `ENCGRP`, and `TIME`) from the DM shown on the previous page. This model allows time variation, behavioral variation and individual heterogeneity in capture probability, yet does so in an efficient and parsimonious (and estimable) manner.

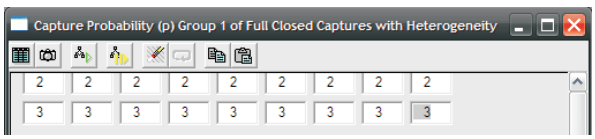
We can use this DM to create additional, reduced parameter models. For example, we could build model $\{N, p_a(t) = c_a(t) = p_b(t) + z = c_b(t) + z\}$ representing capture probability varying through time and additive difference between mixture groups, but with no interaction between p and c over time (no behavior effect). We do this simply by deleting the `ENCGRP` column from the DM.

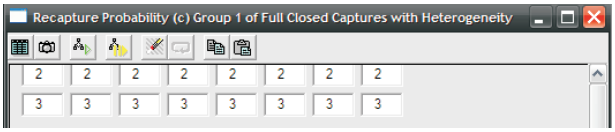
As a final test – how do we modify the DM to match the true generating model, which for these data was model $\{f_0, \pi, p_A = c_A, p_B = c_B\}$? To build this model from our DM, we simply delete (i) all the time columns, (ii) any interactions with time, and (iii) the encounter group column (`ENCGRP`). We delete the encounter group column because we're setting $p = c$. We retain the heterogeneity (mixture) group column (`HETGRP`) since we want to allow for the possibility that encounter probability differs between mixtures (which of course is logically necessary for a mixture model!).

Looking at the real and derived parameter estimates generated by fitting this reduced DM to the data, we see that the estimates ($\hat{\pi} = 0.607, \hat{p}_{\pi_A} = 0.250, \hat{p}_{\pi_B} = 0.754, \hat{N} = 1995.494$). Clearly most of these estimates are very close to the true parameter values used in the generating model. But, what about $\hat{\pi}$? The true value we used in the simulation was $\pi = 0.40$. The estimated value $\hat{\pi} = 0.607$ is simply the complement.

We can confirm that this corresponds to model $\{f_0, \pi, p_A = c_A, p_B = c_B\}$ by comparing the model fit with that from the PIM-based equivalent. We can do this in one of two ways - we can either (i) stay within the '**`Full Likelihood Heterogeneity with pi, p, and c`** ' data type, and build the appropriate PIMs, or (ii) change data type to the simpler '**`Full Likelihood Heterogeneity Pi and p`**', which in fact defaults to our desired model.

If we take the first approach, all we need to do is modify the two encounter probability PIMs as follows, for p and c, respectively:





So, constant over time and no behavior effect (i.e., $p = c$) within mixture group. If you run this model, you'll see that it yields an identical model deviance (555.1792) as the model built using the modified DM (above).

What about changing data types? Well, you might think that you need to restart **MARK**, and begin a new project after first specifying the new data type. In fact, you don't need to - you can simply 'tell' **MARK** that you want to switch data types (something **MARK** lets you do within certain types of models - in this instance, closed population abundance estimators). All you need to do is select '**`PIM | change data type`**' on the main menu bar, and then select '**`Full Likelihood Heterogeneity Pi and p`**' from the resulting popup window. As noted earlier, the default model for this data type is the model we're after - it is simply a reduced parameter version of the full model.

14.7.1. Interpreting π

So, you do an analysis using a closed population heterogeneity abundance model, and derive an estimate of $\hat{\pi}$. Perhaps you've built several such models, and have a model averaged estimate of $\bar{\hat{\pi}}$. So, what do you 'say' about this estimate of $\hat{\pi}$?

Easy answer - *generally nothing*. The estimate of $\hat{\pi}$ is based on fitting a finite mixture model, with a (typically small) number of *discrete* states. When we simulated such data (above), we used a discrete simulation approach - we simply imagined a population where 40% of the individuals had one particular detection probability, and 60% had a different encounter probability. In that case, because the distribution of individuals in the simulated population was in fact discrete, then the real estimate of $\hat{\pi}$ reflected the true generating parameter. However, if in fact the variation in detection probability was (say) continuous, then in fact the estimate of $\hat{\pi}$ reflects a 'best estimate' as to where a discrete 'breakpoint' might be (breaking the data into a set of discrete, finite mixtures). Such an estimate is not interpretable, by and large. Our general advice is to avoid *post hoc* story-telling with respect to $\hat{\pi}$, no matter how tempting (or satisfying) the story might seem.

Closing comment: individual heterogeneity – the bane of abundance estimation

Individual heterogeneity refers to the variation among individual animals in their probability of detection. Most capture-recapture models assume that capture probability is constant across individuals within a group. When individuals vary in the their capture probabilities, the most catchable animals are likely to be caught first and more often. This leads to capture probability being over estimated and abundance being underestimated.

Many attempts have been made to deal with heterogeneity in capture probability over the past 30+ years. No single method has really solved the problem, although several methods are useful. **MARK** allows individual heterogeneity to be approximated with finite mixtures (as above) or with individual covariates (using Huggins conditional likelihood models).

While these approaches can be effective, the single best way to minimize the bias caused by individual heterogeneity is to get p as high as possible – the 'big law' of capture-recapture design. When p is high there is little room for variation and little chance that an individual is not detected. Link (2003) demonstrated that different models of the form of individual heterogeneity can lead to very different estimates of abundance and fit the data equally well. The magnitude of the differences in abundance estimates is related to p; when p is small the differences can be large. Therefore, to have much hope of estimating abundance with little bias, capture probability must be relatively high.

14.8. Misidentification models

The likelihoods and cell probabilities get more complicated when we want to include the possibility of *misidentification* into the cell probabilities. In order to do this we must assume that (i) an individual encountered more than once is correctly identified (i.e., individuals captured on multiple occasions are correctly identified – owing to the greater amount of information gathered on which to base the identification), and (ii) individuals encountered only once may or may not be correctly identified.

First, we consider the closed capture cell probabilities without finite mixtures. We will add the possibility of *misidentification* (where α is the probability of correctly identifying the individual) to the probabilities for a 4-occasion full likelihood closed population capture-recapture model:

history	*cell probability*
1000	$p_1\alpha(1-c_2)(1-c_3)(1-c_4)+p_1(1-\alpha)$
0100	$(1-p_1)[p_2\alpha(1-c_3)(1-c_4)+p_2(1-\alpha)]$
0010	$(1-p_1)(1-p_2)[p_3\alpha(1-c_4)+p_3(1-\alpha)]$
0001	$(1-p_1)(1-p_2)(1-p_3)[p_4\alpha+p_4(1-\alpha)]$
1100	$p_1\alpha c_2(1-c_3)(1-c_4)$
1010	$p_1\alpha(1-c_2)c_3(1-c_4)$
1001	$p_1\alpha(1-c_2)(1-c_3)c_4$
1110	$p_1\alpha c_2c_3(1-c_4)$
1101	$p_1\alpha c_2(1-c_3)c_4$
1011	$p_1\alpha(1-c_2)c_3c_4$
0110	$(1-p_1)p_2\alpha c_3(1-c_4)$
⋮	⋮

In the encounter histories for individuals encountered only once their probability expression is a summation across the two possible ways the history could have occurred; for example, consider history '0100'; captured for the first time, marked and released alive at occasion 2. Conditional on being alive and in the sample (i.e., available for capture) over the entire sampling period, then the probability of observing encounter history '0100' is $(1-p_1)$ (the probability of not being captured at the first occasion), times the sum of (1) the probability the individual was correctly identified and not seen again ($p_2\alpha(1-c_3)(1-c_4)$, or (2) the individual was misidentified and therefore unable to be seen again $p_2(1-\alpha)$.

When misidentification occurs, the constraint that $\hat{N} \geq M_{t+1}$ no longer holds. It is possible that enough animals are misidentified such that the number detected is greater than the number that actually exist in the population. Second, this increase in the numbers of animals supposedly encountered causes the estimated probability of detection to be smaller than it should be. The effect of these two factors is to cause the estimated abundance $\hat{N}$ to be too high.

To account for these problems, the sum $\hat{f}_0 + M_{t+1}$ must be adjusted for mis-identification error, $\hat{\alpha}$:

$$\hat{N} = \hat{\alpha}(\hat{f}_0 + M_{t+1}).$$

Therefore, in these models where misidentification is possible **MARK** presents $\hat{f}_0$ in the real parameter output and $\hat{N}$ in the derived parameter output as it is a function of more than one parameter.

14.8.1. joint heterogeneity and misidentification models

Both the simple and complex heterogeneity models are available for the misidentification closed capture models (i.e., they are available data types). However, incorporation of both misidentification and heterogeneity typically leads to inconclusive results, in that misidentification is somewhat (almost totally) confounded with heterogeneity. Intuitively, misidentification is detected by too many animals only appearing once in the encounter histories. Thus, a large amount of individual heterogeneity may appear as misidentification, and vice versa, misidentification may appear as individual heterogeneity.

So, you can build models with both heterogeneity and misidentification, but there is a very good chance you won't be able to do much with the results.

14.9. Goodness-of-fit

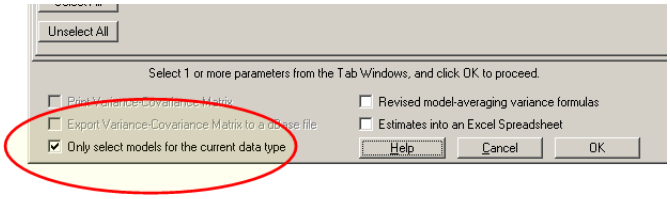
Testing model fit in the closed-population capture-recapture models remains an unresolved issue, even more so than in other capture-recapture model types. A central component of the problem stems from the fact that there is no unique way to compute a saturated model. If one was only concerned about time variation in capture probability, then goodness-of-fit is fairly straightforward. When individual heterogeneity is added into the problem there is an infinite set of possible models for heterogeneity. Thus, no unique goodness-of-fit exists.

Several tests of model assumptions have been developed for the closed-population capture-recapture models (Otis *et al*. 1978:50-67, White *et al*. 1982:77–79). The seven tests examine the fit of specific model forms relative to other specific models or vague alternatives (i.e., the model fails to fit for unspecified reasons). These tests are available in **MARK** through **CAPTURE** by selecting the '**Appropriate**' check box in the **CAPTURE** window. The tests were developed largely as a means of model selection in the absence of another method. Now that **MARK** employs AIC_c as a selection criterion and that it has been shown the model averaged estimates of N have better properties than single-model estimates (Stanley and Burnham 1998), the tests of Otis *et al*. (1978) have fallen out of use.

14.10. Model averaging and closed models

Model averaging is particularly important in the closed models because selecting a single model tends to be especially problematic when a parameter, in this case N, is in the multinomial coefficient. Typically, abundance would be the only parameter for which we're interested in a model averaged estimate. The basic concepts and mechanics of model averaging were introduced in earlier chapters.

To compute a model averaged estimate for abundance, select '**Output | Model Averaging**' then either '**Real**' or '**Derived**' from the menu. Select the appropriate parameter by checking the box from the PIM window that opens. Here, it will be especially important to note the check box in the lower left-hand corner of the model averaging window (highlighted in the red oval, below).



The highlighted 'check box' selects whether model averaging is performed across multiple data types. It is legitimate to model average across data types that are based on the same likelihood, but not across those based on different likelihoods.

What do we mean by 'different likelihoods'? Well, if you look back at the figure at the top of p. 4 in this chapter, you'll see that closed population abundance models are broadly dichotomized based on whether 'f_0, is included in the likelihood' (referred to as 'full likelihood' models), or not (referred to as 'conditional likelihood' or 'Huggins' models). Also recall that within either the full or conditional likelihood models, there are 2 discrete classes of models, depending on whether or not heterogeneity in encounter probability is being modeled using a finite mixture approach. In a moment, we'll discuss why this is important.

First, why is it not legitimate to average over models with different likelihoods? Recall that model averaging is based on an average of parameters over a candidate model set, where the conditional estimates from each individual model are weighted by normalized AIC weights. Also recall that the AIC is calculated as the sum of $-2\ln(\mathcal{L}) + 2K$ parameters. If the underlying models have different likelihoods, then it would clearly not be correct to model average parameters based on AIC weights normalized over those models.

However, while it is not possible to model average between different models based on conditional or unconditional likelihoods, there are two fairly simply approaches which allow you to accommodate the additional the problem of averaging over models with and without finite mixtures. The approach is based on the simple observation that all models are in fact mixture models – but, simply, some of those models have only a single mixture group. These models are, in fact, entirely equivalent conceptually to standard models without mixtures.

We demonstrate model averaging by considering analysis of some simulated data (contained in `N_avg.inp`): true $N = 2{,}000$, 9 sampling occasions. We'll begin by assuming no heterogeneity in p or c, and will use the '**`Full likelihood p and c`**' data type (i.e., f_0 is included in the likelihood) for our analysis of these data.

To start, we'll fit 2 simple models: $\{f_0, p_t = c_t\}$ and $\{f_0, p_t = c_t + z\}$, where the latter model allows for an additive constant z between the two encounter types (recall that this model is equivalent to $M(bt)$, specifying both a 'behavior' effect, and a 'time' effect). While we could use a PIM approach to build model $\{p_t = c_t\}$, for the second additive model, $\{f_0, p_t = c_t + z\}$, we need a DM. So, it is perhaps more efficient to build both models using a DM. Here is the DM for the more general of our 2 candidate models, $\{f_0, p_t = c_t + z\}$:

B1 intcpt	B2 encgrp	Parm	B3 t1	B4 t2	B5 t3	B6 t4	B7 t5	B8 t6	B9 t7	B10 t8	B11 f0
1	1	1:p	1	0	0	0	0	0	0	0	0
1	1	2:p	0	1	0	0	0	0	0	0	0
1	1	3:p	0	0	1	0	0	0	0	0	0
1	1	4:p	0	0	0	1	0	0	0	0	0
1	1	5:p	0	0	0	0	1	0	0	0	0
1	1	6:p	0	0	0	0	0	1	0	0	0
1	1	7:p	0	0	0	0	0	0	1	0	0
1	1	8:p	0	0	0	0	0	0	0	1	0
1	1	9:p	0	0	0	0	0	0	0	0	0
1	0	10:c	0	1	0	0	0	0	0	0	0
1	0	11:c	0	0	1	0	0	0	0	0	0
1	0	12:c	0	0	0	1	0	0	0	0	0
1	0	13:c	0	0	0	0	1	0	0	0	0
1	0	14:c	0	0	0	0	0	1	0	0	0
1	0	15:c	0	0	0	0	0	0	1	0	0
1	0	16:c	0	0	0	0	0	0	0	1	0
1	0	17:c	0	0	0	0	0	0	0	0	0
0	0	18:f0	0	0	0	0	0	0	0	0	1

For DM corresponding to the simpler, nested model $\{f_0, p_t = c_t\}$ – we simply delete the column in the DM corresponding to the 'encounter type' (encgrp).

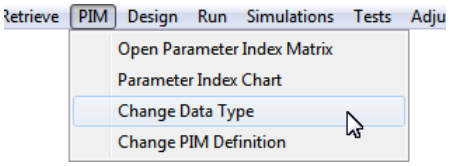
Here are the model fit results for these 2 models:

Model	AICc	Delta AICc	AICc Weight	Model Likelihood	No. Par.	Deviance
{f0,p(t)=c(t)+z}	-1439.0830	0.0000	0.61518	1.0000	11	566.3322
{f0,p(t)=c(t)}	-1438.1447	0.9383	0.38482	0.6255	10	569.2729

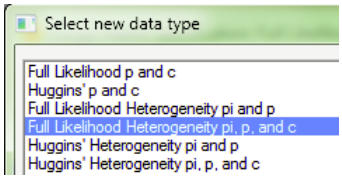
If we stopped here, and model averaged abundance, our model averaged estimate (based on these 2 models) would be $\hat{\bar{N}} = 1{,}996.97$, with an unconditional $\widehat{\text{SE}} = 2.40$.

Now, let's re-analyze these data using a model which assumes heterogeneity in encounter probability, using a finite mixture approach. Our purpose here is to consider model averaging over models with and without mixtures (in other words, based on different data types). In order to do this, we need to build the mixture models within the same '**MARK** project' (since we can only average across models within a given results browser*). To do this, we're going to tell **MARK** that we want to 'change the data type' within our current analysis, from '**Full likelihood p and c**' to something else (specifically, a mixture model).

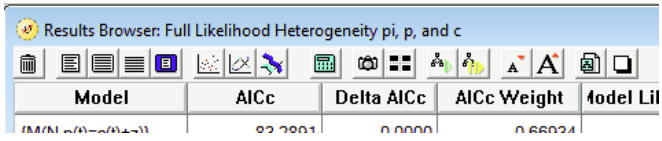
We do this by selecting '**PIM | Change Data Type**':



MARK will then present all the available data types which are consistent with your data, letting you select the one you want to change to. Here, we select '**Full likelihood heterogeneity pi, p and c**':

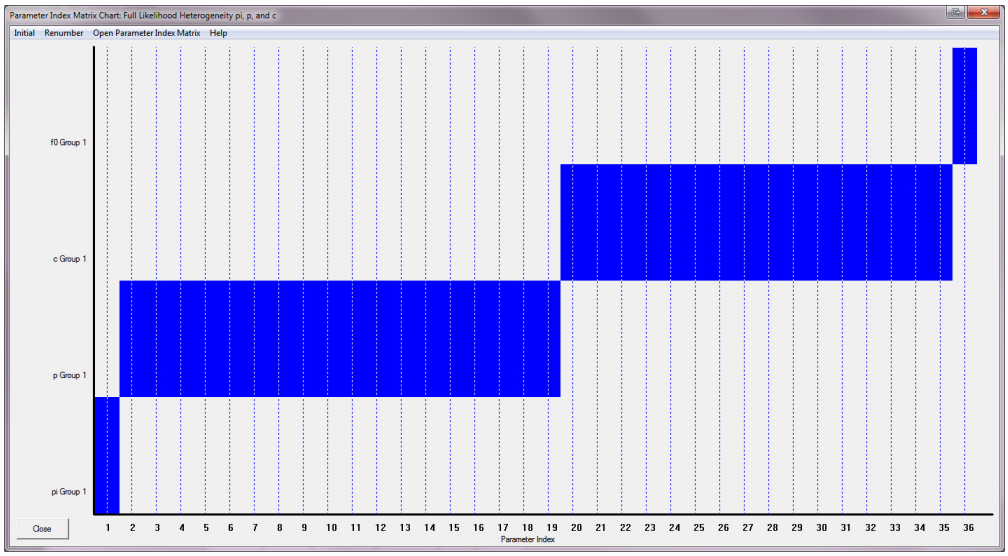


Once we've selected the new data type, **MARK** will ask you how many finite mixtures you want to model. We'll accept the default of 2 mixture groups. **MARK** will then drop you back into the browser – the only indication that the underlying data type has been changed is that the title of the results browser now says 'Full likelihood heterogeneity pi, p, and c'.



* You might wonder if you could start a new project, using mixture models, and then import them, but **MARK** doesn't allow you to import from a different data type. Generally, there is a very good reason for this.

The PIM chart (below) is another indication that the underlying data type has changed.



We see (above) that the default model now has the mixture parameter, π, with full time dependence for both encounter parameters, p and c. Recall from earlier in this chapter that for the heterogeneity models, there are now twice as many p's and c's as you might have expected, given there are 10 occasions represented in the data. This increase represents the parameters for each of the two mixture groups.

For our heterogeneity model, we'll fit model $\{f_0, \pi, p_{A,t} = c_{A,t} + z_A, p_{B,t} = c_{B,t} + z_B\}$ to the data. In other words, $\{p_t = c_t + z\}$, but separately within each of the 2 mixture groups. Here is the DM:

B1 pi	B2 int	B3 encgrp	B4 hetgrp	Parm	B5 t1	B6 t2	B7 t3	B8 t4	B9 t5	B10 t6	B11 t7	B12 t8	B13 f0
1	0	0	0	1:pi	0	0	0	0	0	0	0	0	0
0	1	1	1	2:p	1	0	0	0	0	0	0	0	0
0	1	1	1	3:p	0	1	0	0	0	0	0	0	0
0	1	1	1	4:p	0	0	1	0	0	0	0	0	0
0	1	1	1	5:p	0	0	0	1	0	0	0	0	0
0	1	1	1	6:p	0	0	0	0	1	0	0	0	0
0	1	1	1	7:p	0	0	0	0	0	1	0	0	0
0	1	1	1	8:p	0	0	0	0	0	0	1	0	0
0	1	1	1	9:p	0	0	0	0	0	0	0	1	0
0	1	1	1	10:p	0	0	0	0	0	0	0	0	0
0	1	1	0	11:p	1	0	0	0	0	0	0	0	0
0	1	1	0	12:p	0	1	0	0	0	0	0	0	0
0	1	1	0	13:p	0	0	1	0	0	0	0	0	0
0	1	1	0	14:p	0	0	0	1	0	0	0	0	0
0	1	1	0	15:p	0	0	0	0	1	0	0	0	0
0	1	1	0	16:p	0	0	0	0	0	1	0	0	0
0	1	1	0	17:p	0	0	0	0	0	0	1	0	0
0	1	1	0	18:p	0	0	0	0	0	0	0	1	0
0	1	1	0	19:p	0	0	0	0	0	0	0	0	0
0	1	0	1	20:c	0	1	0	0	0	0	0	0	0
0	1	0	1	21:c	0	0	1	0	0	0	0	0	0
0	1	0	1	22:c	0	0	0	1	0	0	0	0	0
0	1	0	1	23:c	0	0	0	0	1	0	0	0	0
0	1	0	1	24:c	0	0	0	0	0	1	0	0	0
0	1	0	1	25:c	0	0	0	0	0	0	1	0	0
0	1	0	1	26:c	0	0	0	0	0	0	0	1	0
0	1	0	1	27:c	0	0	0	0	0	0	0	0	0
0	1	0	0	28:c	0	1	0	0	0	0	0	0	0
0	1	0	0	29:c	0	0	1	0	0	0	0	0	0
0	1	0	0	30:c	0	0	0	1	0	0	0	0	0
0	1	0	0	31:c	0	0	0	0	1	0	0	0	0
0	1	0	0	32:c	0	0	0	0	0	1	0	0	0
0	1	0	0	33:c	0	0	0	0	0	0	1	0	0
0	1	0	0	34:c	0	0	0	0	0	0	0	1	0
0	1	0	0	35:c	0	0	0	0	0	0	0	0	0
0	0	0	0	36:f0	0	0	0	0	0	0	0	0	1

If we fit this model to the data, and add the results to the browser (below), we see that this new 'heterogeneity model' gets roughly 84% of the support in the data among our 3 candidate models:

Model	AICc	Delta AICc	AICc Weight	Model Likelihood	No. Par.	Deviance
{f0,pi,p(A,t)=c(A,t)+z,p(B,t)=c(B,)+z}	-1443.3683	0.0000	0.83981	1.0000	13	558.0413
{f0,p(t)=c(t)+z}	-1439.0830	4.2853	0.09855	0.1173	11	566.3322
{f0,p(t)=c(t)}	-1438.1447	5.2236	0.06164	0.0734	10	569.2729

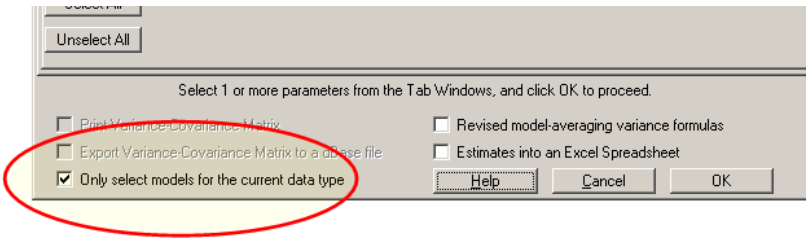
But, our interest here concerns model averaging. If at this point, having just fit the heterogeneity model, $\{N, \pi, p_{A,t} = c_{A,t} + z_A, p_{B,t} = c_{B,t} + z_B\}$, we run through the (by now) familiar mechanics of model averaging for N, we would see only one model reported in the model averaging output:

```
                              model averaging
    Estimates only for data type Full Likelihood Heterogeneity pi, p, and c

                            Derived Parameter 1
Model                                    Weight    Estimate      Standard Error
---------------------------------------- -------   -------------- --------------
{f0,pi,p(A,t)=c(A,t)+z,p(B,t)=c(B,)+z}   1.00000   1998.3164945   2.8151126
---------------------------------------- -------   -------------- --------------
Weighted Average                                   1998.3164945   2.8151126
Unconditional SE                                                  2.8151126
```

Why only one model, and not all three? Simple – at present there is only one model in the browser based on the 'currently active' data type (i.e., full likelihood with 2 finite mixtures). **MARK** knows that the other 2 models in the current model set were constructed using the a different data type ('full likelihood without mixtures'), and thus doesn't try to average over them. Alternatively, if you select (by right-clicking and retrieving) either of the other two models we constructed using the **'Full likelihood p and c'** data type (i.e., $\{f_0, p_t = c_t\}$ or $\{f_0, p_t = c_t + z\}$), and then model average, the model averaging will be based on these 2 models only (since they share a common data type).

Note: Not only is **MARK** smart enough to recognize which models in the browser are based on the same data type, but it is also smart enough to re-calculate AIC weights during the averaging to include only those models with the common (active) likelihood structure. So, the model averaged estimated is correctly reported as $\hat{\bar{N}} = 482.15$, with an unconditional $\widehat{\text{SE}} = 185.76$ (identical to what we reported earlier for these 2 models, before we changed the data type).

Back to the problem at hand. Remember at the outset of this section we alerted you to the default (selected) option in the model averaging procedure in **MARK**, to '**only select models for the current data type**' (as circled in red, below).



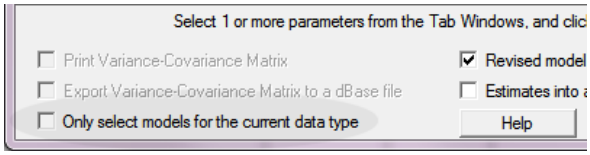
This is the option which 'tells **MARK**' to average only over models of the current data type.

However, our apparent inability to model average over the complete model set represented in the browser seems unfortunate, since we might imagine a full candidate model set with and without heterogeneity models, over which we'd like to derive a model averaged estimate for abundance, $\hat{N}$. What can we do?

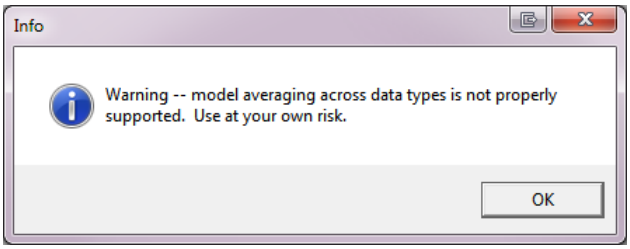
There are 2 related approaches you can adopt to average over all 3 models – both of which are based on the same assumption. For either approach, the key conceptual step is to realize that *any* model constructed using the **'Full likelihood p and c'** data type is simply a heterogeneity model constructed using the **'Full likelihood heterogeneity pi, p, and c'** data type, with one important change – fixing $\pi = 1$. (Similarly, any model constructed using '**Huggins p and c**' is simply a '**Huggins heterogeneity p and c**' model, again after fixing $\pi = 1$). If you think about it for a moment, this should make sense – the **'Full likelihood p and c'** data type is simply a heterogeneity model with only one mixture group (i.e., where $\pi = 1$). So, you could, if you wanted to, force the **'Full likelihood heterogeneity pi, p, and c'** data type to fit models for the **'Full likelihood p and c'** data type, simply by fixing the mixture parameter π to 1. We'll consider this approach in a moment.

The quickest approach to handling model averaging in this case is to 'tell **MARK**' to ignore the fact that, structurally, there are two different data types in the browser. We can do this here because, in fact, the **'Full likelihood p and c'** data type is simply a full likelihood heterogeneity model where π is fixed to 1. In other words, although the models represent two different data types, but have the same underlying likelihood structure. In fact, one data type is equivalent to the other, subject to a particular constraint on one of the parameters (i.e., fixing $\pi = 1$).

So, we run through the mechanics of model averaging, except that this time, we 'turn off' the option to restrict averaging to only models of the current data type, by unchecking the appropriate check-box, as shown below:



Now, when you uncheck this option, **MARK** will respond with the following rather ominous warning message when you try to average over models:



Here, 'use at your own risk' means 'make sure you know what you're doing...'. In this instance, we'll assume our underlying logic is correct, and so we can proceed with the final steps of model averaging abundance N.

Here are the estimates:

model averaging

Derived Parameter 1

Model	Weight	Estimate	Standard Error
{f0,pi,p(A,t)=c(A,t)+z,p(B,t)=c(B,)+z}	0.83981	1998.3164945	2.8151126
{f0,p(t)=c(t)+z}	0.09855	1997.5435131	2.4970104
{f0,p(t)=c(t)}	0.06164	1996.0406631	1.9051863
Weighted Average		1998.1000288	2.7276734
Unconditional SE			2.7972971

We see that now, **MARK** averages over all 3 of the models in the browser – the model averaged estimate for abundance is $\hat{\hat{N}}$ = 1,998.10, with an unconditional $\widehat{\text{SE}}$ = 2.80

However, are these estimates correct? Did we in fact 'know what we were doing' when we overrode **MARK**'s warning about averaging over data types? Was our underlying logic that in fact these models have the same underlying likelihood structure correct? We can prove to ourselves that 'we got things right' (and confirm that **MARK** has given us the correct estimates using the preceding approach) by (i) reconstructing the model set using the same data type for all three models, and (ii) manually fixing $\pi = 1$ for two of them. While this is easy enough in principle, in practice this approach will require some thought, since you're going to need to think through carefully which columns in the **'Full likelihood heterogeneity pi, p, and c'** data type DM you need to keep, or modify, when you are reducing the number of heterogeneity groups to 1 (i.e., single mixture group).

To start, have another look at the DM for model $\{f_0, \pi, p_{A,t} = c_{A,t} + z_A, p_{B,t} = c_{B,t} + z_B\}$, shown on p. 31. Notice that there is a column for 'hetgrp', to account for the 2 mixture groups in this model. If we want to force this model to be equivalent to a model without heterogeneity, without switching the underlying data type, we need to do 2 things: (1) delete the 'hetgrp' column from the DM, and (2) fix $\pi = 1$ before starting the numerical estimation run.

Go ahead and delete the 'hetgrp' column from the DM. What is the model represented by this DM? If you look closely, and think about it a bit, you'll realize that without the 'hetgrp' column, you're left with model $\{p_t = c_t + z\}$. Go ahead and run this model – call it 'f0,pi=1,p(t)=c(t)+z' (we'll use 'pi=1' in the model name to indicate we built this model using only a single mixture group). Remember to fix $\pi = 1$ before starting the numerical estimation.

When finished, add the results to the browser:

Model	AICc	Delta AICc	AICc Weight	Model Likelihood	No. Par.	Deviance
{f0,pi,p(A,t)=c(A,t)+z,p(B,t)=c(B,)+z}	-1443.3683	0.0000	0.76447	1.0000	13	558.0413
{f0,p(t)=c(t)+z}	-1439.0830	4.2853	0.08971	0.1173	11	566.3322
{f0,pi=1,p(t)=c(t)+z}	-1439.0830	4.2853	0.08971	0.1173	11	566.3322
{f0,p(t)=c(t)}	-1438.1447	5.2236	0.05611	0.0734	10	569.2729

We see that the deviances for model 'f0,pi=1,p(t)=c(t)+z' and model 'f0,p(t)=c(t)+z' are identical (meaning, they are the same model!).

Next, how would we build model $\{p_t = c_t\}$, using the heterogeneity model approach? Simple – in addition to deleting the 'hetgrp' column, we now also delete the 'encgrp' column (leaving only 'pi', 'incpt', the time columns ('t1' → 't9'), and N. Go ahead and delete the 'encgrp' column, fix $\pi = 1$, and add the results to the browser (call this model 'pi=1,p(t)=c(t)').

Model	AICc	Delta AICc	AICc Weight	Model Likelihood	No. Par.	Deviance
{f0,pi,p(A,t)=c(A,t)+z,p(B,t)=c(B,)+z}	-1443.3683	0.0000	0.72386	1.0000	13	558.0413
{f0,p(t)=c(t)+z}	-1439.0830	4.2853	0.08494	0.1173	11	566.3322
{f0,pi=1,p(t)=c(t)+z}	-1439.0830	4.2853	0.08494	0.1173	11	566.3322
{f0,p(t)=c(t)}	-1438.1447	5.2236	0.05313	0.0734	10	569.2729
{f0,pi=1,p(t)=c(t)}	-1438.1447	5.2236	0.05313	0.0734	10	569.2729

Again, we see that fits for model '`N,pi=1,p(t)=c(t)`' and model '`N,p(t)=c(t)`' are identical (meaning, once again, that they are the same model!).

OK, now for the big moment. We've proven to ourselves that we can build models for the **'Full likelihood p and c'** data type using the **'Full likelihood heterogeneity pi, p, and c'** data type, simply by fixing $\pi = 1$, and making appropriate modifications to the DM (paying particular attention to terms involving the 'heterogeneity group' column). So, in fact, we could have built all 3 candidate models ($\{p_t = c_t\}, \{p_t = c_t + z\}$ and $\{\pi, p_{A,t} = c_{A,t} + z_A, p_{B,t} = c_{B,t} + z_B\}$), using the **'Full likelihood heterogeneity pi, p, and c'** data type – meaning, a single common data type. Meaning, we can model average over all 3 models without overriding the default option in **MARK** that prevents averaging over models built using different data types.

Go ahead and delete models '`p(t)=c(t)+z`' and '`p(t)=c(t)`' from the browser, leaving only those models built using the **'Full likelihood heterogeneity pi, p, and c'** data type (i.e., all 3 models in the browser are based on the same underlying data type):

Model	AICc	Delta AICc	AICc Weight	Model Likelihood	No. Par.	Deviance
{f0,pi,p(A,t)=c(A,t)+z,p(B,t)=c(B,)+z}	-1443.3683	0.0000	0.83981	1.0000	13	558.0413
{f0,pi=1,p(t)=c(t)+z}	-1439.0830	4.2853	0.09855	0.1173	11	566.3322
{f0,pi=1,p(t)=c(t)}	-1438.1447	5.2236	0.06164	0.0734	10	569.2729

Now, the big moment – go ahead and derive a model averaged estimate for $\hat{\hat{N}}$, based on these 3 models – without unchecking the '**Only select models for the current data type**' (since these models are all of the same data type):

```
                              model averaging
    Estimates only for data type Full Likelihood Heterogeneity pi, p, and c

                            Derived Parameter 1
Model                                    Weight    Estimate      Standard Error
---------------------------------------- -------   -------------- --------------
{f0,pi,p(A,t)=c(A,t)+z,p(B,t)=c(B,)+z}   0.83981   1998.3164945   2.8151126
{f0,pi=1,p(t)=c(t)+z}                    0.09855   1997.5434147   2.4970007
{f0,pi=1,p(t)=c(t)}                      0.06164   1996.0408300   1.9052311
---------------------------------------- -------   -------------- --------------
Weighted Average                                   1998.1000294   2.7276752
Unconditional SE                                                  2.7972924
```

Using this approach, the model averaged estimate for abundance is $\hat{\hat{N}} = 1{,}998.10$, with an unconditional $\widehat{\text{SE}} = 2.80$, which are identical to the estimates we derived earlier.

Given the preceding, there is a fair argument to be made that you could only use the **'heterogeneity pi, p, and c'** data types (for either the full or conditional likelihoods), since it allows you to model average over all the candidate models. However, keeping track of 'encounter groups' and 'heterogeneity

groups' does require more work to get things right. As long as you understand what you're doing, simply forcing **MARK** to average over both data types is decidedly quicker. But, remember – you can only average over models with a common likelihood structure: full likelihood (with and without mixtures), or Huggins conditional likelihood (with and without mixtures).

14.10.1. Estimating CI for model averaged abundance estimates

The usual (simplest) approach to estimating the confidence interval for a given parameter makes use of asymptotic variances, covariances - typically, these can be generated from the information matrix for models with maximum likelihood estimates (this is discussed elsewhere).

However, there is a basic problem with applying this 'classical' approach to estimates of abundance - specifically, the classical approach requires asymptotic normality of point estimates $\hat{N}$, and this assumption is frequently not met for any number of reasons.

An alternative approach is to focus on the number of animals that are not caught (f_0), where $f_0 = (N - M_{t+1})$ (this relation was introduced earlier in this chapter). On the assumption that this quantity follows a log-normal distribution (which has been generally confirmed by various authors), then lower and upper CI interval bounds for $\hat{N}$ are given by*

$$\left[M_{t+1} + (\hat{f}_0/C), M_{t+1} + (\hat{f}_0 \times C)\right]$$

where

$$\hat{f}_0 = \hat{N} - M_{t+1}$$

$$C = \exp\left\{1.96\left[\ln\left(1 + \frac{\widehat{\mathrm{var}}(\hat{N})}{\hat{f}_0^2}\right)\right]^{1/2}\right\}$$

Note that since $\hat{N} = M_{t+1} + \hat{f}_0$, then $\widehat{\mathrm{var}}(\hat{N})$ is exactly the same as the variance of $\hat{f}_0$, because M_{t+1} is a known constant.

As such,

$$\frac{\widehat{\mathrm{var}}(\hat{N})}{\hat{f}_0^2} = \frac{\widehat{\mathrm{var}}(\hat{f}_0)}{\hat{f}_0^2}$$

$$= \widehat{\mathrm{CV}}(\hat{f}_0)^2$$

Commonly in these kinds of calculations, the square of the CV (coefficient of variation) of f_0 is embedded in the formula.

It is important to note that the lower bound of this confidence interval cannot be smaller than M_{t+1}, but the upper bound frequently is larger than the upper bounds computed with the information matrix under the assumption of normality. This is the approach used by **MARK** to derive the CI for $\hat{N}$ (regardless of whether N is a derived or real parameter).

Now, how do we handle the calculation of the CI for the model averaged estimate of abundance, $\bar{\hat{N}}$?

* There is a typographical error in the equation for C in the Williams, Nichols & Conroy book (p. 304, section 14.2.4). The version presented here is correct.

From Buckland *et al*. (1997), the estimated unconditional (i.e., model averaged) variance $\widehat{\text{var}}(\hat{\bar{\theta}})$, calculated over models $\{M_1, M_2, \ldots, M_R\}$ is given as

$$\widehat{\text{var}}(\hat{\bar{\theta}}) = \sum_{i=1}^{R} w_i \left(\widehat{\text{var}}(\hat{\theta}_i \mid M_i) + (\hat{\theta}_i - \hat{\bar{\theta}})^2 \right), \quad \text{where } \hat{\bar{\theta}} = \sum_{i=1}^{R} w_i \hat{\theta}_i$$

Here, the w_i are the Akaike weights (Δ_i) scaled to sum to 1. The subscript i refers to the ith model. The value $\hat{\bar{\theta}}$ is a weighted average of the estimated parameter θ over R models ($i = 1, 2, \ldots, R$).

This estimator of the *unconditional* variance is clearly the sum of 2 components: (i) the *conditional* sampling variance $\widehat{\text{var}}(\hat{\theta}_i | M_i)$ (i.e., conditional on model M_i), and (ii) a term for the variation in the estimates across the R models $(\hat{\theta}_i - \hat{\bar{\theta}})^2$. The sum of these terms is then merely weighted by the Akaike weights w_i.

Thus, the unconditional standard error would be given as

$$\widehat{\text{SE}}(\hat{\bar{\theta}}) = \sqrt{\widehat{\text{var}}(\hat{\bar{\theta}})}$$

OK - given all this, back to the original question - how do you estimate the confidence interval for model averaged abundance estimates? We'll demonstrate the mechanics by means of a worked example. Suppose you fit 3 different models to some closed capture data, where $M_{t+1} = 47$. We'll call these models M_1, M_2 and M_3 - assume they're all based on the same likelihood form, and differ only in parameter structure. Here is a tabulation of the relevant results of fitting these models to the data:

model	QAIC_c	w_i	$\hat{N}$	$\widehat{\text{var}}(\hat{N})$
M_1	272.5064	0.63665	59.1161	63.8798
M_2	273.6392	0.36134	58.1909	56.0402
M_3	284.0209	0.00201	56.4051	35.0438

Now, we first need to calculate the unconditional variance of $\hat{N}$. Since our model averaged estimate of $\hat{\theta}$ is given as

$$\hat{\bar{\theta}} = \sum_{i=1}^{R} w_i \hat{\theta}_i$$

then $\hat{\bar{N}}$ is given as

$$\hat{\bar{N}} = \sum_{i=1}^{R} w_i \hat{N}_i$$

$$= (0.63665 \times 59.1161) + (0.36134 \times 58.1909) + (0.00201 \times 56.4051)$$

$$= 58.7763$$

and

$$\begin{aligned}\widehat{\text{var}}(\hat{\bar{N}}) &= \sum_{i=1}^{R} w_i\left(\widehat{\text{var}}(\hat{N}_i \mid M_i) + (\hat{N}_i - \hat{\bar{N}})^2\right) \\ &= 0.63665\left[63.8798 + (59.1161 - 58.7763)^2\right] + 0.36134\left[56.0402 + (58.1909 - 58.7763)^2\right] \\ &\quad + 0.00201\left[35.0438 + (56.4051 - 58.7763)^2\right] \\ &= (40.7426 + 20.3734 + 0.0817) \\ &= 61.1977\end{aligned}$$

In fact, **MARK** handles the calculation of the unconditional variance for you – you would simply need to take the reported unconditional SE and square it to get the unconditional variance. But you need to calculate the CI by hand.

To do so, we first calculate

$$C = \exp\left\{1.96\left[\ln\left(1 + \frac{\widehat{\text{var}}(\hat{\bar{N}})}{\hat{f}_0^2}\right)\right]^{1/2}\right\}$$

Since $M_{t+1} = 47$ for this data set, and since $\hat{\bar{N}} = 58.7763$, then

$$\begin{aligned}\hat{\bar{f}}_0 &= \hat{\bar{N}} - M_{t+1} \\ &= 58.7763 - 47 \\ &= 11.7763\end{aligned}$$

and thus

$$\begin{aligned}C &= \exp\left\{1.96\left[\ln\left(1 + \frac{\widehat{\text{var}}(\hat{\bar{N}})}{\hat{f}_0^2}\right)\right]^{1/2}\right\} \\ &= \exp\left\{1.96\left[\ln\left(1 + \frac{61.1977}{(11.7763)^2}\right)\right]^{1/2}\right\} \\ &= 3.2707\end{aligned}$$

Last step. Now that we have a value for C, we can derive the 95% CI as

$$[47 + (11.7763/3.2707), 47 + (11.7763 \times 3.2707)] = [50.601, 85.517]$$

OK, this seems like a lot of work, but in this particular example, it was necessary. If we had simply used the 'automatic' model averaging in **MARK**, the CI reported by **MARK** for $\hat{\bar{N}}$ was [43.443, 74.116]. There is clearly a major problem with this CI, since the lower bound is less than M_{t+1} ($43.443 < 47$). Clearly, this makes no sense whatsoever. In contrast, the CI we derived 'by hand' does not bound M_{t+1}. Not only was the reported lower-limit of the CI too low, but the upper limit was as well.

Now, in the preceding example, there was an *obvious* 'problem' with the simple model-averaged CI for $\hat{\bar{N}}$ reported by **MARK**. However, even if the lower bound of the reported CI is $\geq M_{t+1}$, don't take this as evidence that the reported CI is correct.

For example, consider fitting models $\{N, p(.) = c(.)\}$ and $\{N, p(.), c(.)\}$ (corresponding to model M_0

and M_b in the older, 'Otis' notation) to the 'Carothers A' data set (found in the `\examples` subdirectory created when you installed **MARK**).

Here is a tabulation of the relevant results of fitting these models to the data:

model	QAIC_c	w_i	$\hat{N}$	$\widehat{\text{var}}(\hat{N})$
$\{N, p(.) = c(.)\}$	-99.7370	0.63460	368.128	212.944
$\{N, p(.), c(.)\}$	-98.6330	0.36540	392.480	1234.986

If we had used the model averaging option in **MARK**, the model averaged estimate is $\hat{\bar{N}} = 377.027$, and the reported 95% CI is [324.292, 429.761]. For this data set, $M_{t+1} = 283$, so, in one sense at least, the reported CI for the model average abundance estimate *seems* reasonable, since the lower limit of the CI is greater than M_{t+1} (i.e., 324.292 > 283). How does the reported CI compare with the one derived using the calculations presented above?

Again, we start by first deriving an estimate of the variance of the model averaged abundance:

$$\begin{aligned}\widehat{\text{var}}(\hat{\bar{N}}) &= \sum_{i=1}^{R} w_i\left(\widehat{\text{var}}(\hat{N}_i \mid M_i) + (\hat{N}_i - \hat{\bar{N}})^2\right) \\ &= 0.63460\left(212.944 + (368.128 - 377.027)^2\right) \\ &\quad + 0.36540\left(1234.986 + (392.480 - 377.027)^2\right) \\ &= 723.910\end{aligned}$$

Note that if we were to fit these models in **MARK**, the unconditional SE for the model averaged abundance would be reported as 26.9045 – if we square this value, we get $(26.9045)^2 = 723.901$. Again, the unconditional SE – and thus the variance – reported by **MARK** is correct (i.e., you do not need to calculate the SE – or variance – by hand. We are simply demonstrating the underlying calculations). However, the CI as reported by **MARK** is not correct – this, you need to do by hand.

As in the first example, we first calculate

$$C = \exp\left\{1.96\left[\ln\left(1 + \frac{\widehat{\text{var}}(\hat{\bar{N}})}{\hat{f}_0^2}\right)\right]^{1/2}\right\}$$

Since $M_{t+1} = 283$ for this data set, and since $\hat{\bar{N}} = 377.027$, then

$$\begin{aligned}\hat{f}_0 &= \hat{\bar{N}} - M_{t+1} \\ &= 377.027 - 283 \\ &= 94.027\end{aligned}$$

Thus,

$$C = \exp\left\{1.96\left[\ln\left(1 + \frac{723.910}{(94.027)^2}\right)\right]^{1/2}\right\}$$

$$= 1.733$$

Final step. Now that we have a value for C, we can construct the 95% CI around the model averaged estimate $\hat{\bar{N}} = 377.027$ as

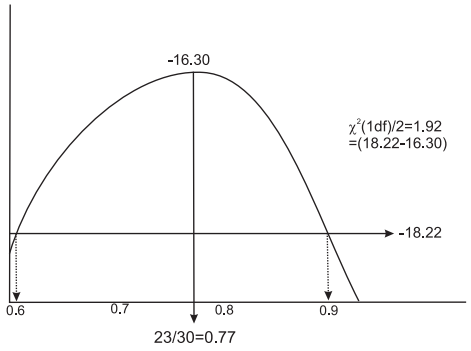
$$\begin{aligned}&\Big[283 + (94.027/1.733), 283 + (94.027 \times 1.744)\Big] \\ &= [337.26, 445.94]\end{aligned}$$

Recall that if we had used the model averaging option in **MARK**, the reported model averaged 95% CI was [324.292, 429.758]. Again, these reported lower- and upper-limits of the CI are both different than the ones we just calculated 'by hand'. The general recommendation, then, is to calculate the 95% CI for the model averaged abundance 'by hand', using the procedure outlined above.

begin sidebar

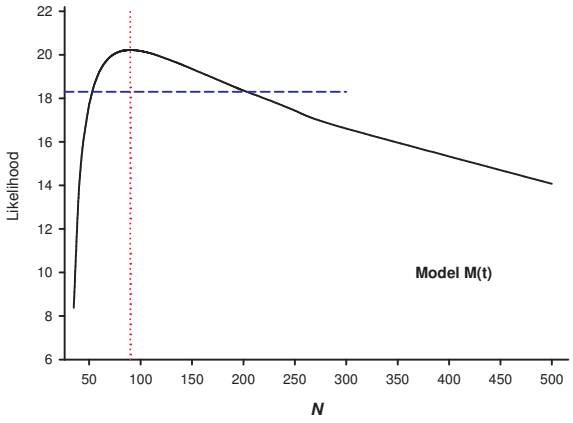
Profile confidence intervals - careful!

In chapter 1, we introduced the profile likelihood approach to constructing confidence intervals. Typically, you take the value of the log likelihood at the maximum (−16.30 in the example, below), add 1.92 to it (preserving the sign), and look to see where the line corresponding to this sum (−18.22 = −[16.30 + 1.92]) intersects with the *profile* of the log likelihood function. The two intersection points of this line and the profile correspond to the upper- and lower-bounds of the CI.



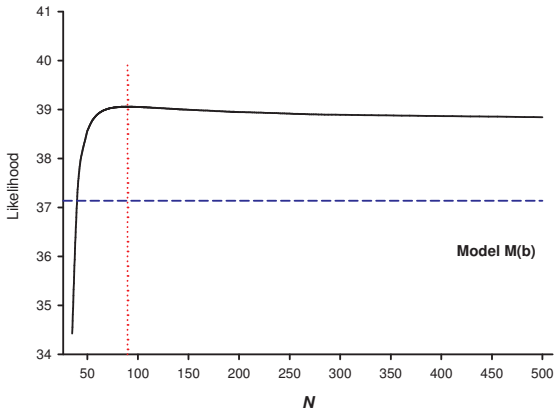
For closed abundance estimators, there is need to be cautious in using profile likelihoods to generate CI, having to do with the fact that abundance estimates are not [0, 1] bounded parameters, but are minimally bound at 0. The maximum bound (if in fact one exists) is determined by the likelihood. There are situations for some closed models where the upper bound of the likelihood profile $\rightarrow \infty$.

For example, take the likelihood for model $\{N, p(t) = c(t)\}$ (i.e., model M_t) fit to some data:



We see that for model $\{N, p(.) = c(.)\}$, the likelihood profile rises to the MLE (vertical dotted line), and then falls, such that the horizontal dashed line corresponding to the MLE−1.92 intersects the likelihood at 2 points (which represent the two bounds of the 95% CI).

However, now consider model $\{N, p(.), c)(.)\}$ (i.e., model M_b)



Here, the likelihood rises, but then never falls to <2 units from the MLE - and, as such, there is no upper bound for the profile likelihood!

end sidebar

14.11. Parameter estimability in closed models

It is important to examine the real parameter results to see if $p_t = 1.0$ and $\hat{N} = M_{t+1}$. This would indicate that the model you constructed was not estimable. Be careful – incorrectly built models may appear very good in terms of AIC_c. If you don't know what M_{t+1} is for a particular data set, it can be found in the full model output labeled as '`M(t+1):`'.

In addition, it has been noted several times that a constraint must be placed on p_t in order to properly estimate N. It is straightforward to demonstrate that an estimate of p_t is necessary to get an estimate of N. We've already done it once. We'll do it again here to make sure you don't forget.

Consider the following estimator of N from a $t = 3$ occasion capture-recapture study,

$$\hat{N} = \frac{M_{t+1}}{1 - (1 - \hat{p}_1)(1 - \hat{p}_2)(1 - \hat{p}_3)}.$$

Now if $\hat{p}_3 = 1$, then the denominator in the estimator above equals 1. Thus, the estimate of $\hat{N} = M_{t+1}$.

Let's consider the estimability of the p's, now that we know we need $\hat{p}_t$ to get $\hat{N}$. The first p is estimable because we have information in the subsequent capture occasions about the proportion of marked and unmarked animals captured. This goes for each p until we get to p_t. On the last occasion, there are no future occasions from which to pull information. Thus, we must place a constraint of p_t. The constraint can be in the form of modeling p_t as a function of previous p's or as a function of the recaptures, or by constraining estimates to be functions of one or more covariates. Recall that constraining parameters as linear function of a covariate can often 'solve' identifiability issues.

14.12. Other applications

Closed population capture-recapture models have been used for other applications beyond estimating the number of individuals in a population. There is a natural extension to estimating the number of species in an area. In this case, encounter histories represent detections of species rather than individuals. Heterogeneity in detection probability among species is virtually guaranteed.

Closed capture-recapture models and modifications thereof are widely used in human demography. There they are typically referred to as multiple list sampling. Several lists containing people from a population of interest, for example drug users in a city, act as sampling occasions. Individuals are matched across lists to estimate abundance.

The closed population capture-recapture models underpin the secondary sampling periods in a robust design (Kendall *et al*. 1997; see Chapter 15). It is therefore essential to understand the closed captures models in order to fully understand the robust design

14.13. Summary

Despite a seemingly simple goal, estimating abundance can be quite difficult. The closed capture-recapture models contain numerous, subtle complications. **MARK** offers a framework for a variety of models addressing different assumptions, compares models and most importantly model averages estimated abundance.

An additional advantage of **MARK** is the ability to combine data from multiple study sites with ease. It is too often argued in the ecological literature that capture-recapture is not useful because the sample size at any one trapping grid is too small. Through the use of groups, **MARK** allows data from multiple grids to be used to jointly estimate detection probability. While this may bias the estimate of N somewhat for each individual grid, it remains a far better solution than using minimum number known alive as an index. Moreover, **MARK** handles all of the covariances among the N's estimated from common data.

14.14. References

Boulanger, J., G. C. White, B. N. McLellan, J. Woods, M. Proctor, and S. Himmer. 2002. A meta-analysis of grizzly bear DNA mark-recapture projects in British Columbia, Canada. *Ursus*, **13**, 137-152.

Carothers, A. D. 1973. Capture-recapture methods applied to a population with known parameters. *Journal of Animal Ecology*, **42**, 125-146.

Huggins, R. M. 1989. On the statistical analysis of capture experiments. *Biometrika*, **76**, 133-140.

Kendall, W. L., J. D. Nichols, and J. E. Hines. 1997. Estimating temporary emigration using capture-recapture data with Pollock's robust design. *Ecology*, **78**, 563-578.

Link, W. A. 2004. Nonidentifiability of population size from capture-recapture data with heterogeneous detection probabilities. *Biometrics*, **59**, 1123-1130.

Lukacs, P. M., and K. P. Burnham. 2005. Estimating population size from DNA-based closed capture-recapture data incorporating genotyping error. *Journal of Wildlife Management*, **69**, 396-403.

Otis, D. L., K. P. Burnham, G. C. White, and D. R. Anderson. 1978. Statistical inference from capture data on closed animal populations. *Wildlife Monographs*, **62**.

Pledger, S. 2000. Unified maximum likelihood estimates for closed capture-recapture models using mixtures. *Biometrics*, **56**, 434-442.

Stanley, T. R., and K. P. Burnham. 1999. A closure test for time-specific capture-recapture data. *Environmental and Ecological Statistics*, **6**, 197-209.

Stanley, T. R., and K. P. Burnham. 1998. Information-theoretic model selection and model averaging for closed-population capture-recapture studies. *Biometrical Journal*, **40**, 475-494.

White, G. C., D. R. Anderson, K. P. Burnham, and D. L. Otis. 1982. Capture-recapture and removal methods for sampling closed populations. Los Alamos National Laboratory Publication LA-8787-NERP. Los Alamos, NM.

CHAPTER 15

The 'robust design'

William Kendall, *USGS Colorado Cooperative Fish & Wildlife Research Unit*

Changes in population size through time are a function of births, deaths, immigration, and emigration. Population biologists have devoted a disproportionate amount of time to models that assume immigration and emigration are non-existent (or, not important). However, modern thinking suggests that these effects are potentially (perhaps generally) quite important. For example, metapopulation dynamics are not possible without immigration and emigration in the subpopulations. A model which allows the estimation of emigration and immigration to a population is therefore of considerable utility. In this chapter, we consider Pollock's *robust design*, an approach which will allow us considerable flexibility in estimating a very large number of important demographic parameters, including estimates of emigration and immigration. As you might imagine, such a model is bound to be more complicated than most (if not all) of the models we've previously considered, but it brings more biological reality to the analysis of population dynamics.

15.1. Decomposing the probability of subsequent encounter

We begin by considering the probabilistic pathway that links two events – the initial capture, marking and live release of an individual, and its subsequent re-encounter (for the moment, we'll focus on live encounters). We know by now that we can represent such an individual with the encounter history '11'. An individual that we mark and release but do not encounter on the subsequent sampling occasion would have the encounter history '10'. Back in Chapter 1, we motivated the need for estimating encounter probability by considering the utility of measures of *return rate*. You might recall that 'return rate'* is *not* a robust measure of survival. Why? Well, recall from Chapter 1 that 'return rate' is, at minimum, the product of two events: (1) the probability of surviving from the time of initial mark and release to some future sampling occasion, and (2) the probability that the individual is encountered on that sampling occasion, conditional on being alive. Because the 'return rate' is in fact the product of two different probabilities, this makes it difficult (and frequently impossible) to determine if differences in 'return rate' are due to differences in the probability of survival, the probability of encounter, or both. To solve this problem, we introduced models which explicitly account for encounter probability, such that potential differences in survival probabilities can be determined.

* Recall the 'return rate' is simply the proportion of individuals marked and released at some occasion that are encountered on a subsequent occasion; in other words, 'return rate' is simply $x(11)/[x(11)+x(10)]$, where $x(11)$ is the number of individuals marked and encountered on a subsequent occasion, and $x(10)$ is the number marked and not encountered on a subsequent occasion.

© Cooch & White (2014)

In fact, our treatment of 'return rate' (both in the preceding paragraph, and in Chapter 1) is incomplete. It is incomplete because in fact 'return rate' is the product of more than two parameters – it is the product of at least 4 lower-level parameters. We can illustrate this dependence graphically, using a 'fate diagram', as indicated in Fig. (15.1):

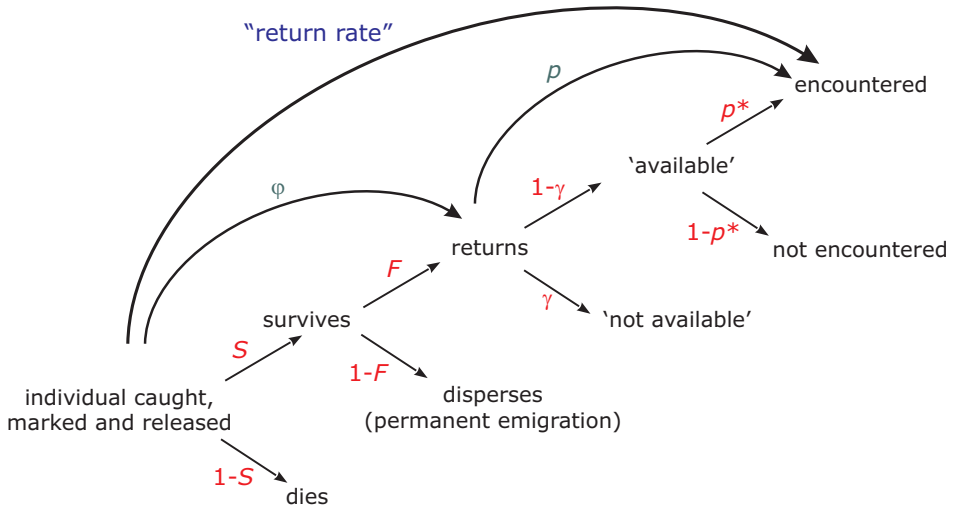


Figure 15.1: *Basic fate diagram indicating the decomposition of 'return rate' into component transition parameters:* $\boldsymbol{S}$ *(probability of surviving from release occasion* i *to subsequent sampling period* $i+1$*),* $\boldsymbol{F}$ *(probability that, conditional on surviving, that individual does not permanently leave (e.g., by permanent emigration) the population being sampled (i.e., the super-population; see Kendall 1999),* $\boldsymbol{(1-\gamma)}$ *(the probability that conditional on being alive, and in the super-population, that the individual is available to be encountered), and* $\boldsymbol{p^*}$ *(the probability that an individual is encountered, conditional on being alive, in the super-population, and available for encounter). The arcs indicate the underlying structure of apparent survival probability (*$\boldsymbol{\varphi = S \times F}$*), apparent encounter probability (*$\boldsymbol{p = (1-\gamma) \times p^*}$*), and 'return rate' (*$\boldsymbol{= S \times F \times (1-\gamma) \times p^*}$*).*

Starting at the lower left-hand corner of Fig. (15.1), we see that an individual animal is caught, marked and released alive at occasion i. Then, there are several 'events' which determine if the individual is encountered alive at a subsequent sampling occasion $i + 1$. First, the animal must survive – we use the parameter S to denote survival. Clearly, the probability of the animal not surviving is given by the complement probability $(1 - S)$. This much should be pretty obvious.

Next, conditional on surviving, a marked individual is potentially available for subsequent encounter if it remains in the '*super-population*' (the larger population from which we are sampling). We use the parameter F to indicate the probability of fidelity of the marked individual to the super-population. We note that the fidelity parameter F was first introduced in Chapter 9, in the context of joint live encounter-dead recovery analysis. The complement $(1 - F)$ is the probability that the animal has *permanently* left the super-population, e.g., by dispersing, and would thus not be available for subsequent *live* encounter in a sample drawn from this super-population under any circumstances.

Next, conditional on remaining in the super-population (with probability F), we introduce the concept of 'availability'. It's perhaps easiest to introduce this idea based on a simple biological example. Suppose we're dealing with a bird species, where only breeding individuals are found at the breeding site where we conduct our encounter sampling. Clearly, then, only breeding individuals are 'available' for encounter, whereas non-breeding individuals would be 'unavailable'. We model the probability of an individual being *unavailable* using the parameter γ (such that the probability of being *available* is

given by its complement $1 - \gamma$). Note that in most instances, the availability of a marked individual for encounter is conditional, varying from occasion to occasion (e.g., in some years, a marked individual breeds, and is thus available, whereas in other years, the same individual does not breed, and is thus unavailable). As such, we generally refer to the parameter γ as defining the probability that the marked individual has or has not *temporarily emigrated* from the study area. So, γ might be considered as the probability that the marked individual has temporarily emigrated from the study area. In fact, we'll see shortly that the γ parameter can be interpreted in more than one way.

Finally, conditional on surviving, remaining in the super-population, and being available for encounter, the marked individual is encountered live with probability p^*. Here, we use the asterisk '$*$' to differentiate what we will refer to as the 'true' encounter probability (p^*) from the 'apparent' encounter probability (p). The use of the familiar p to indicate apparent encounter probability is intentional, since it forces us to acknowledge that the familiar p parameter estimated in most models focused on live encounter data is in fact a 'function' of the true encounter rate, but is not true encounter rate in and of itself (except under very specific circumstances).

To make this clear, let's write out the following expression for 'return rate'. As noted earlier (and in Chapter 1), 'return rate' is in fact the product of two separate events – survival and encounter. But, we also noted that this simple definition is incomplete. It's incomplete, because it is more strictly correct to say that 'return rate' is the product of the *apparent* survival probability and the *apparent* encounter probability. If we let R represent return rate, and use φ and p to represent apparent survival rate and encounter probability, respectively, then we can write

$$R = (\varphi \times p)$$

Now, considering Fig. (15.1), we see that apparent survival (φ) is itself a product of true survival (S), and fidelity (F). This should make sense – the probability that an animal marked and released alive at occasion i will be encountered alive in the study area at occasion $i + 1$ requires that the animal survives (with probability S), and remains in the super-population (with probability F; if it permanently emigrates, then it will appear 'dead', since permanent emigration and mortality are confounded). So, $\varphi = SF$. Similarly, apparent encounter probability p is the product of the probability that the animal is available for encounter (with probability $1 - \gamma$), and the true detection probability p^* (which is the probability of detection, given availability, or presence). So, $p = (1 - \gamma)p^*$. Thus, we write

$$\begin{aligned} R &= \text{'apparent survival probability'} \times \text{'apparent encounter probability'} \\ &= (\varphi \times p) \\ &= (SF) \times ([1 - \gamma]p^*) \end{aligned}$$

Now, in several previous chapters, we simply decomposed 'return rate' R into apparent survival φ and apparent encounter probability p. The challenge, then, is to further decompose φ and p into their component pieces. In Chapter 9, we considered use of combined live encounter-dead recovery data to decompose φ. Recall that dead recovery data provides an estimate of true survival rate S, whereas live encounter data yields estimates of apparent survival probability φ. Since $\varphi = (SF)$, then an *ad hoc* estimate of F is given as $\hat{F} = (\varphi/S)$. The formal likelihood-based estimation of $\hat{F}$ (described by Burnham, 1993) is covered in detail in Chapter 9.

What about the decomposition of apparent encounter probability p? We see from Fig. (15.1) that $p = (1 - \gamma)p^*$. Following the logic we followed in the preceding paragraph to derive an *ad hoc* estimator for F, we see that $\hat{p}^* = \hat{p}/(1 - \hat{\gamma})$, and $\hat{\gamma} = 1 - (\hat{p}/\hat{p}^*)$; estimates of both the true encounter probability, and the 'availability' probability may be of significant interest.

15.2. Estimating γ: the classical 'live encounter' RD

The problem, then, is how to derive an estimate of either p^* or γ? Recall that we can generate an estimate for p (apparent encounter probability) using our standard live encounter CJS models. But, where do we get estimates of γ, and p^*? Are any of them estimated by any estimation model we've considered so far?

Well, if you think back to Chapter 14 (on closed population estimators), you might recall that one of the parameters estimated is in fact p^*. Now, in Chapter 14, we didn't refer to the parameter using the p^* notation, but with a few moments of thought, you should see they are essentially the same thing (well, not quite – recall that closed capture models estimate two different 'types' of encounter probability – p and c – we'll deal with these details later). In a closed population, there is neither entry or exit of individuals (i.e., N is a constant). As such, your estimate of the encounter probability is not conditional on presence or availability, since (by definition for a closed population) the marked individuals must be there. So, the estimate of p from a closed population model allows you to derive an estimate of p^* (given $n > 1$ occasions, $p^* = 1 - [(1 - p_1)(1 - p_2) \ldots (1 - p_n)]$).

OK, fine, but why is this important? It's important because *if* you have an estimate of apparent encounter probability p, and *if* you have an estimate of true encounter probability p^*, then you can derive an *ad hoc* estimate of γ as $\hat{\gamma} = 1 - (\hat{p}/\hat{p}^*)$.

Now, for the 'big leap forward'. To derive the estimate of γ, we need an estimate of p (which we can get from standard open, live encounter CJS models), and p^* (which we can get from standard closed estimates). Can we derive both estimates from the same data set (based on samples from the same population)?

The answer (as first described by Ken Pollock) is 'yes' – by application of what has been described as the *robust design*. The robust design model is a *combination* of the Cormack-Jolly-Seber (CJS) (Cormack 1964, Jolly 1965, Seber 1965) live recapture model and the closed capture models. The model is described in detail by Kendall *et al*. (1997, 1995) and Kendall and Nichols (1995), and is represented schematically in standard ('classical') form in Fig. (15.2):

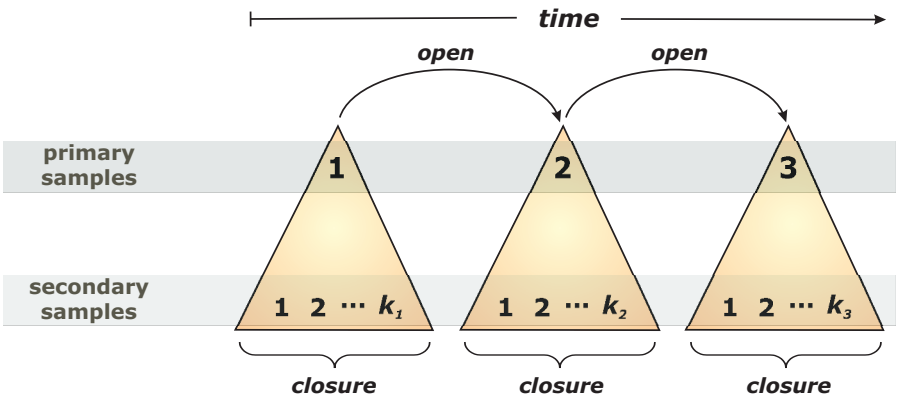


Figure 15.2: *Basic structure of 'classical' Pollock's robust design.*

The key difference from the standard CJS model considered in several earlier chapters is that instead of just one capture occasion between survival intervals, multiple (>1) capture occasions are used. These occasions are close together in time – so close that it allows you to (in general) assume that the populations are closed while these samples are being taken (i.e., no mortality or emigration occurs during these short time intervals). In fact, Pollock pointed out that in many cases data were being

collected in this way anyway (e.g., small mammal sampling might be conducted in groups of 5-7 consecutive trapping days). The closed encounter occasions are termed *secondary* trapping occasions, and each primary trapping session can be viewed as a closed capture survey.

The power of this model is derived from the fact that, in addition to providing estimates of abundance ($\hat{N}$), the probability that an animal is captured at least once in a trapping session can be estimated from the data collected during the session using capture-recapture models developed for closed populations (Chapter 14). The longer intervals between *primary* trapping sessions allows estimation of survival, temporary emigration from the trapping area, and immigration of marked animals back to the trapping area. The interval between primary sampling sessions is sufficiently long that gains (birth and immigration) and losses (death and emigration) to the population can occur. This contrasts with secondary samples (within the primary sampling session), where the interval between samples is sufficiently short that the population is effectively closed to gains and losses.

Recall that what we're after are estimates of both p and p^*, from which we can derive an estimate of γ. The relationship of the various parameters to the standard robust design is shown in Fig. (15.3):

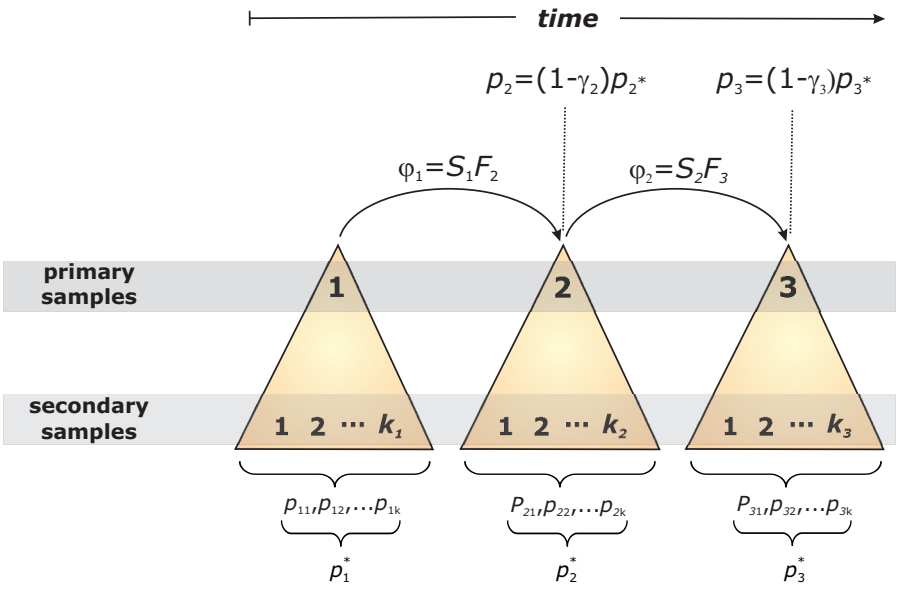


Figure 15.3: *Relationship of key parameters to basic structure of Pollock's robust design.*

For each secondary trapping session (i), the probability of first capture p_{ij} and the probability of recapture c_{ij} are estimated (where j indexes the number of trapping occasions within the session), along with the number of animals in the population that are on the trapping area N_i. For the intervals between trapping sessions (i.e., between primary sessions, when the population is open), the probability of apparent survival φ_i (= $S \times F$), and the apparent encounter probability p are estimated.

It is clear from Fig. (15.3) that it should be possible to derive estimates of γ. In the absence of extra information (specifically, dead recovery data, or the equivalent), partitioning apparent survival φ into component elements S and F is not feasible using the classical robust design (which is based entirely on live encounters at a single location). We will deal with extensions to the classic robust design later in this chapter.

15.3. The RD extended – temporary emigration: γ' and γ''

Earlier we introduced the parameter γ as the probability that the individual was 'unavailable' for encounter at some particular primary sampling session. Kendall *et al*. (1995a, 1997) extended the simple (classical) parameterization of the robust design in terms of parameter γ by introducing two different parameters: γ' and γ'' (read as 'gamma-prime' and 'gamma-double-prime', respectively). Basically, these two new parameters are defined as follows:

parameter	*definition*
γ'_i	the probability of being *off* the study area, unavailable for capture during primary trapping session (i) given that the animal *was not* present on the study area during primary trapping session ($i-1$), and survives to trapping session (i).
γ''_i	the probability of being *off* the study area, unavailable for capture during the primary trapping session (i) given that the animal *was* present during primary trapping session ($i-1$), and survives to trapping session (i).

Now, these are perhaps more difficult to 'wrap your brain around' than they might first appear. You need to read the definitions carefully.

First, we distinguish between the 'observable' (i.e., potentially available for encounter at time i) and 'unobservable' (i.e., potentially unavailable for encounter at time i) parts of the population of interest. The 'superpopulation' (i.e., the target population of interest) is the sum of the 'observable' and 'unobservable' individuals. The γ define the probability of movement between the 'observable' and 'unobservable' states. The basic relationship between γ' and γ'' is shown in Fig. (15.4):

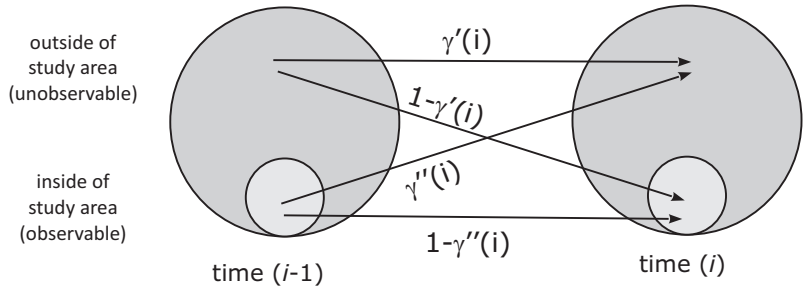


Figure 15.4: *Relationships between γ' and γ''. The larger circle represents the range of the super-population. The smaller circle (light grey) represents the part of the superpopulation that is available for encounter (i.e., in the study area), whereas the darker part of the larger circle represents individuals unavailable for encounter (i.e., temporarily outside the study area).*

Start with the parameter γ''_i. It is the probability that given that you were available at time $(i-1)$, that you are not available now at time (i). In other words, γ'' is the probability of an individual that is available for encounter at time $(i-1)$ temporarily emigrating between time $(i-1)$ and (i), such that it is not available for encounter at time (i). Thus, $(1-\gamma'')$ is the probability of being in the study area at time (i), given that it was also in the sample at time $(i-1)$. As indicated in Fig. (15.4), the parameter γ''_i is the probability of temporarily emigrating *from* the sample between sampling occasions $(i-1)$ and (i),

and its complement $(1 - \gamma_i'')$ is the probability of remaining in the sample between sampling occasions $(i - 1)$ and (i).

What about parameter γ'? Again, consider Fig. (15.4) - γ' is the probability that given that an individual was *not* in the sample at time $(i-1)$, that is also *not* present (i.e., not in the sample) at time (i). In effect, γ' is the probability of remaining outside the sample (if you prefer, 'fidelity' to being outside the sample). Thus, $(1 - \gamma')$ is the probability that an individual which *was* out of the sample at time $(i - 1)$ *enters* the sample between time $(i - 1)$ and time (i) - i.e., *return rate of temporary emigrants*.

Indexing of these parameters (as indicated in Fig. 15.4) follows the notation of Kendall *et al*. (1997). Thus, γ_2'' applies to the interval before the second primary trapping session. It is important to note that not all parameters are estimable (either because of *logical* constraints, or *statistical confounding*). For example, γ_2' is not estimated because there are no marked animals outside the study area at primary trapping session 2 that were also outside the study area at time 1 (because they could not have been marked otherwise). In general, for a study with k primary sessions, (i) $S_1, S_2, \ldots, S_{k-1}$, (ii) $p_{ij}, i = 1 \ldots k, j = 1 \ldots k_i$, (iii) $\gamma_3', \gamma_4', \ldots \gamma_{k-1}'$, and (iv) $\gamma_2'', \gamma_3'', \ldots \gamma_{k-1}''$ are estimable. General issues of estimability of various parameters is discussed elsewhere (below).

15.3.1. γ parameters and multi-state notation

If these parameters are still confusing, note the similarity of Fig. (15.4) to multi-state models introduced in Chapter 10. In fact, this temporary emigration model is a special case of a multi-state model with two states. Defining state **O** to be the study area (**O**; observable) and state **U** to be off the study area (**U**; unobservable), then $\gamma_3'' = \psi_2^{OU}$ and $\gamma_3' = \psi_2^{UU}$. The basic relationship between the ψ parameters and the 'observable' and 'unobservable' states is shown in Fig. (15.5).

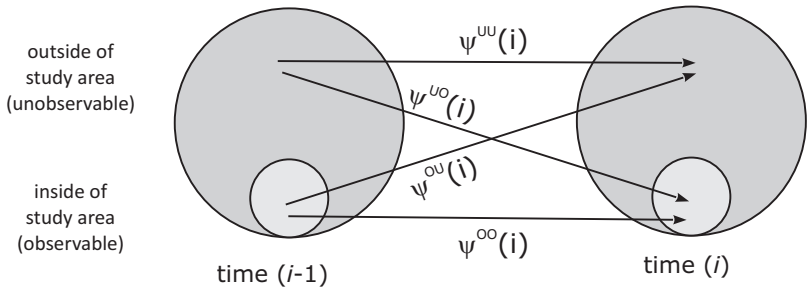


Figure 15.5: *Multi-state (ψ) probabilities of transition between 'observable' and 'unobservable' states.*

If you compare Fig. (15.5) with Fig. (15.4) for a few moments, you should recognize that

$$\begin{aligned} \gamma' &\equiv \psi^{UU} \\ 1 - \gamma' &\equiv \psi^{UO} \\ \gamma'' &\equiv \psi^{OU} \\ 1 - \gamma'' &\equiv \psi^{OO} \end{aligned}$$

In fact, you could, with a bit of work, perform a 'typical' single sampling location robust design problem as a multi-state problem with two states – you would simply fix $S = 1$ and $\psi^{OU} = 0$ in the closed periods (modeling the encounter probability only). In the '**`Closed Robust Design Multi-state`**'

and '**Open Robust Design Multi-state**' options in **MARK**, which we describe later in this chapter, we abandon the use of the 'γ notation' altogether. Although those models are more flexible, the models using γ that we are discussing here are much simpler to set up.

15.3.2. illustrating the extended model: encounter histories & probability expressions

To illustrate the mechanics of fitting the classical robust design model, assume a simple case with 3 primary trapping sessions, each consisting of 3 secondary trapping occasions. The encounter history in its entirety is viewed as 9 live capture occasions, but with unequal spacing. Thus, the encounter history might be viewed as

$$1\ 1\ 1 \longrightarrow 1\ 1\ 1 \longrightarrow 1\ 1\ 1$$

where the '$\rightarrow$' separates the primary trapping sessions. The probability that an animal is captured at least once during a trapping session is defined as p_i^* (see Chapter 14), and is estimated as

$$p_i^* = 1 - \left[(1 - p_{i1}) \times (1 - p_{i2}) \times (1 - p_{i3})\right]$$

That is, the probability of not seeing an animal on trapping occasion j is $(1 - p_{ij})$ for $j = 1, 2,$ and 3. The probability of *never* seeing the animal during trapping session i is

$$(1 - p_{i1}) \times (1 - p_{i2}) \times (1 - p_{i3})$$

so therefore, the probability of seeing the animal at least once during the trapping session is 1 minus this quantity. Note that the p_{ij} are estimated as with the closed capture models (Chapter 14).

To illustrate the meaning of the emigration (γ_i'') and immigration (γ_i') parameters, suppose the animal is captured during the first trapping session, not captured during the second trapping session, and then captured during the third trapping session. One of many encounter histories that would demonstrate this scenario would be (where spaces in the encounter history separate primary sampling sessions, but which would not appear in an actual encounter history):

```
010   000   111
```

which, if pooled over secondary samples within primary samples, would be equivalent to the encounter history '101'. The probability of observing this 'pooled' encounter history can be broken down into 2 parts. First, consider the portion of the probability associated with the *primary* intervals. This would be

$$\varphi_1\varphi_2\left[\gamma_2''(1 - \gamma_3') + (1 - \gamma_2'')(1 - p_2^*)(1 - \gamma_3'')\right]p_3^*$$

The product in front of the first bracket $[\varphi_1\varphi_2]$ is the probability that the individual survived from the first primary trapping session to the third primary trapping session. Because we encountered it alive on the third occasion (i.e., at least once during the three secondary trapping sessions during the third primary session), we know the individual survived both intervals (this is a logical necessity, obviously).

The complicated-looking term in the brackets represents the probability that the individual was not captured during the second trapping session. The first product within the brackets $[\gamma_2''(1 - \gamma_3')]$ is the probability that the individual emigrated between the first 2 primary trapping sessions (γ_2''), and

then immigrated back onto the study area during the interval between the second and third trapping sessions $[1-\gamma_3']$. However, a second possibility exists for why the animal was not captured, i.e., that it remained on the study area and just was not captured. The term $[1-\gamma_2'']$ represents the probability that the individual 'remained on the study area'. The term $[1-p_2^*]$ represents individuals 'not captured'. The final term $[1-\gamma_3'']$ represents the probability that the individual remained on the study area so that it was available for capture during the third trapping session.

The second portion of the cell probability for the preceding encounter history (p_3^*) involves the estimates of p_i^*, and is thus just the closed capture model probabilities.

15.3.3. Random (classical) versus Markovian temporary emigration

The probability of movement between 'availability states' can be either *random*, or *Markovian*. If the former (random), the probability of moving between availability states between primary occasions i and $i+1$ is independent of the previous state of the system, whereas for Markovian movement, the probability of moving between availability states between primary occasions i and $i+1$ is conditional on the state of the individual at time $i-1$. Note that random movement is essentially what was assumed under the classical robust design model discussed earlier (i.e., the RD model based on γ, and not parameterized in terms of γ' and γ'').

To provide identifiability of the parameters for the '*Markovian emigration*' model (where an animal 'remembers' that it is off the study area) when parameters are time-specific, Kendall *et al*. (1997) stated that γ_k'' and γ_k' need to be set equal to γ_t'' and γ_t', respectively, for some earlier period. Otherwise these parameters are confounded with S_{t-1}. They suggested setting them equal to γ_{k-1}'' and γ_{k-1}', respectively, but it really should depend on what makes the most sense for your situation. This confounding problem goes away if either movement or survival is modeled as constant over time.

To obtain the '*Random emigration*' model, set $\gamma_i' = \gamma_i''$. This constraint is perhaps not intuitively obvious. The interpretation is that the probability of emigrating during an interval is the same as the probability of staying away (i.e., the probability of not immigrating). Biologically, the probability of being in the study area during the current trapping session is the same for those animals previously in and those animals previously out of the study area during the previous trapping session. The last survival parameter, S_{k-1}, is also not estimable under the time-dependent model unless constraints are imposed. That is, the parameters γ_k'', γ_k', and S_{k-1} are all confounded. Setting the constraints $\gamma_{k-1}'' = \gamma_k''$ and $\gamma_{k-1}' = \gamma_k'$, for example, makes the resulting 3 parameters estimable. Or, you could forgo the constraint – in that case, you would simply ignore the estimates of S_{k-1}, γ_k', and γ_k''. Estimates of the remaining parameters would be unbiased.

The null model for both the random and Markovian models is the '*No emigration*' model. To obtain the '*No emigration*' model, you simply set all the γ parameters to zero. If all the γ_i'' are set to zero, then the γ_i' must all be set to zero also, because there are no animals allowed to emigrate to provide a source of immigrants back into the population.

To make the distinction between the random (classical) and Markovian temporary emigration robust design models clearer, consider the cell probability expressions for the following encounter history:

```
110  000  010  111
```

Here, we have 4 primary trapping occasions, and 3 secondary trapping occasions per primary occasion. If we considered only primary occasions, the encounter history for this individual would be '`1011`'. The individual was marked and released on the first secondary occasion within the first primary sampling occasion, and then seen again on the second secondary occasion within that first primary period. The

individual was not seen at all during any of the secondary samples during the second primary sampling occasion. The individual was seen once – on the second of the secondary sampling occasions – during the third primary sampling occasions, and was seen on all of the secondary sampling occasions during the final primary sampling period.

Again, what is key here is the second primary sampling occasion – during the second primary occasion, the individual was not seen at all. This might occur in one of three ways. First, the individual could have died – we assume only live encounters are possible. However, since the individual was seen alive at least once on a subsequent primary sample, then we clearly cannot assume that the '000' secondary encounter history on the second primary occasion reflects death of the individual. However, there are two other possibilities:

1. the individual could be alive and in the sample, but simply 'missed' (i.e., not encountered)

or, alternatively

2. the individual could have temporarily emigrated from the sampling region between primary occasion 1 and primary occasion 2, such that it is unavailable for encounter during primary occasion 2.

We have to account for both possibilities when constructing the probability statements. The following table shows the probability expressions corresponding to this encounter history

```
110  000  010  111
```

for both the Markovian and random temporary emigration models:

model	*probability*
Markovian	$\varphi_1\gamma''_2\varphi_2(1-\gamma'_3)p^*_3\varphi_3(1-\gamma''_4)p^*_4$ $+\varphi_1(1-\gamma''_2)(1-p^*_2)\varphi_2(1-\gamma''_3)p^*_3\varphi_3(1-\gamma''_4)p^*_4$
random	$\varphi_1\gamma_2\varphi_2(1-\gamma_3)p^*_3\varphi_3(1-\gamma_4)p^*_4$ $+\varphi_1(1-\gamma_2)(1-p^*_2)\varphi_2(1-\gamma_3)p^*_3\varphi_3(1-\gamma_4)p^*_4$

Look at these expressions carefully. Make sure you understand the distinction between the random and Markovian temporary emigration models, and how the various constraints needed for identifiability affect the probability expressions. For example, notice that for the random temporary emigration model, the probability expression corresponding to the encounter history is parameterized in terms of γ – no 'gamma-prime' (γ') or 'gamma-double-prime' (γ'') parameters. Why? Well, recall that in order to obtain the '*Random emigration*' model, you set $\gamma'_i = \gamma''_i$ (i.e., simply set both parameters equal to some common parameter γ_i).

Now, let's step through each expression, to make sure you see how they were constructed. Let's start with the Markovian emigration expression. Note that the probability expression for both models is written in two pieces (separated by the '+' sign). These two pieces reflect the fact that we need to account for the two possible ways by which we could achieve the '000' encounter history for the second primary sampling occasion: either (i) the individual was not available to be sampled (with probability

γ_2''; in other words, it was in the sample at primary occasion 1, and left the sample at primary occasion 2, such that it was unavailable for encounter), or (ii) was in the sample during primary sampling occasion 2, with probability $(1 - \gamma_2'')$, but was simply missed (i.e., not encountered). So, let's consider the first part of the probability expression. Clearly, φ_1 indicates the individual survived from primary occasion $1 \rightarrow 2$ – we know this to be true. The γ_2'' term indicates the possibility that the individual temporarily emigrated from the sample between occasions 1 and 2, such that it was unavailable for encounter during primary sampling occasion 2. Then, φ_2, since the individual clearly survives from occasion 2 to occasion 3. Then, conditional on having temporarily emigrated at occasion 2, we need to account for the re-entry (immigration) back into the sample at occasion 3, with probability $(1 - \gamma_3')$. This is logically necessary since the individual was encountered at least once during primary sampling occasion 3. Next, φ_3, since the individual clearly survives from occasion 3 to 4. Finally, the individual stays in the sample (since it was encountered), with probability $(1 - \gamma_4'')$, and was encountered with probability p_4^*.

Now, the second term of the expression (after the '+' sign) is similar, with one important difference – in the second term, we account for the possibility that the individual stayed in the sample between primary sampling occasion 1 and 2 with probability $(1 - \gamma_2'')$, and was not encountered during any of the secondary samples during primary sampling occasion 2 with probability $(1 - p_2^*)$.

For the random emigration model, the expressions are the same, except we've eliminated the 'primes' for the γ terms (we note that we could, with a bit of algebra, reduce both expressions to simpler forms – especially the expression for random emigration. However, leaving the expressions in 'expanded' form makes the logic of how the expressions were constructed more obvious).

15.3.4. Alternate movement models: no movement, and 'even flow'

While in the preceding we focussed on contrasting random and Markovian movement models, it is clear that both need to be tested against an explicit null of '*No movement*'. For this null model, we assume that individuals that are 'observable' are always 'observable' over all sampling occasions. Similarly, individuals which are 'unobservable' remain unobservable over all sampling occasions. We construct the '*No movement*' fairly easily, by simply setting the γ's to 1 (unobservable individuals remain unobservable) and γ''s to 0 (observable individuals remain observable).* Unless you have compelling evidence to the contrary, it is always worth including a '*No movement*' model in your candidate model set.

Another, somewhat more subtle model, is what we might call an '*Even flow*' model. In the '*Even flow*' model, we are interested in whether the probability of moving from 'observable' at time i to 'unobservable' at time $i+1$ is the same as the probability of moving from 'unobservable' to 'observable' over the same time interval. In other words, $(1 - \gamma') = \gamma''$. Note that the 'even flow' model says only that the *per capita* probability of moving to the alternate state over some interval is independent of the originating state at the start of the interval. Be sure you understand the distinction between the '*Even flow*' model and the '*Random movement*' and '*Markovian movement*' models. In the '*Random movement*' model we set $\gamma' = \gamma''$, which means that the probability of an individual being unobservable at time $i + 1$ is independent of whether or not it was 'observable' at time i. As noted earlier, the interpretation is that the probability of *emigrating* during an interval is the same as the probability of staying away (conditional on already being 'unobservable' at the start of the interval). For the '*Markovian movement*' model, we allow for movement rates to differ as a function of whether the individual is 'observable' or 'unobservable' – the only constraints we apply to the γ parameters in the Markovian model are

* Practically speaking, it would not matter if you fixed γ' to 1 or 0. Since the model does not consider movement of marked animals outside the study area, γ' never enters the likelihood and therefore it doesn't matter whether you fix it to 0 or 1. However, setting $\gamma' = 1$ for a 'no movement' model is logically more consistent with Fig. (15.4).

necessary to ensure identifiability. Contrast this with the '*Even flow*' model, where we enforce an equality constraint between entry and exit from a given state over the interval. We will leave it to you do decide which of these models are sufficiently 'biological plausible' to consider including in your candidate model set.

The following table (15.1) summarizes some of the constraints which are commonly used to specify (and in some case, make identifiable) the 4 model types we've discussed so far ('*No movement*', '*Random movement*', '*Markovian movement*', and '*Even flow*').

Table 15.1: *Parameter constraints for standard model types using classical closed RD (γ) parameterization.*

model	*constraint*
no movement	$\gamma' = 1, \gamma'' = 0$
random movement	$\gamma' = \gamma''$
Markovian movement	$\gamma'_k = \gamma'_{k-1}$ $\gamma''_k = \gamma''_{k-1}$
'even flow'	$\gamma'' = (1 - \gamma')$

15.4. Advantages of the RD

Advantages of the robust design alluded to above include

1. estimates of p_i^*, and thus N_i and recruitment are less biased by heterogeneity in capture probability (specifically, if you use heterogeneity models within season; see Chapter 14)
2. temporary emigration can be estimated assuming completely random, Markovian, or temporarily trap dependent availability for capture (Kendall and Nichols 1995, Kendall *et al*. 1997)
3. If temporary emigration does not occur, abundance, survival, and recruitment can be estimated for all time periods (e.g., in a 4-period study, half the parameters are inestimable using the JS method; Kendall and Pollock 1992).
4. Precision tends to be better using the formal robust design models of Kendall *et al*. (1995), which include the model described above with $\gamma'' = \gamma' = 0$.
5. Because there is information on capture for the youngest catchable age class, estimation of recruitment into the second age class can be separated into *in situ* recruitment and immigration when there are only 2 identifiable age classes. Using the classic design (i.e., one capture session per period of interest), 3 identifiable age classes are required (Nichols and Pollock 1990).
6. The robust design's 2 levels of sampling allow for finer control over the relative precision of each parameter (Kendall and Pollock 1992).

15.5. Assumptions of analysis under the RD

For the most part, the assumptions under the robust design are a combination of the assumptions for closed-population methods and the JS method.

1. Under the classical robust design (as first described by Pollock, and subsequently extended by Kendall and colleagues; hereafter, we refer to this as the *closed robust design*), the population is assumed closed to additions and deletions across all secondary sampling occasions within a primary sampling session. Kendall (1999) identified 3 scenarios where estimation of p_i^* would still be unbiased when closure was violated.

    a. If movement in and out of the study area is completely random during the period, then the estimator for p_i^* remains unbiased. The other 2 exceptions require that detection probability vary only by time and might apply most with migratory populations.

    b. If the entire population is present at the first session within a period but begins to leave before the last session, then the estimator is unbiased if detection histories are pooled for all sessions that follow the first exit from the study area. If the exodus begins after the first session this creates a new 2-session detection history within period.

    c. Conversely, if sampling begins before all animals in the population have arrived but they are all present in the last session, then all sessions up to the point of first entry should be pooled.

2. Temporary emigration is assumed to be either completely random, Markovian, or based on a temporary response to first capture.
3. Survival probability is assumed to be the same for all animals in the population, regardless of availability for capture. This is a strong assumption, especially in the Markovian availability case.

15.6. RD (closed) in MARK – some worked examples

OK, enough of the background for now. Let's actually use the closed robust design in **MARK**. We'll begin with a very simple example which can be addressed using only PIMs and the PIM chart, followed by a more complex model requiring modification(s) of the design matrix.

15.6.1. Closed robust design – simple worked example

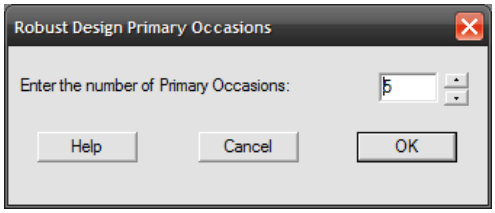
We'll demonstrate the 'basics' using some data simulated under a '*Markovian movement*' model. The data (contained in `rd_simple1.inp`) consist of 3,000 individuals in a study area, some of which are captured, marked and released alive. Each of the 5 primary sampling sessions consisted of 3 secondary samples. So, in total, $(5 \times 3) = 15$ sampling occasions. For our simulation, we assumed that survival between primary periods varied over time: $S_1 = 0.7, S_2 = 0.8, S_3 = 0.9, S_4 = 0.8$. Within each year, we assumed that the true model for encounters during the secondary samples was model $\{N, p(.) = c(.)\}$ (i.e., model M_0 – see Chapter 14). We used $p_{11\rightarrow13} = 0.5, p_{21\rightarrow33} = 0.6, p_{41\rightarrow43} = 0.5$ and $p_{51\rightarrow53} = 0.5$. (*Note*: setting $p_{11} = p_{12} = p_{13} = 0.5$ implies that p_1^*, the probability of being captured at least once in primary period 1, is $p_1^* = 1 - (1 - 0.5)(1 - 0.5)(1 - 0.5) = 0.875$. If you notice, the total number

of individuals captured at least once in primary session 1 in the simulated data set is 2,619, which is close to the expected value of $3{,}000 \times 0.875 = 2{,}625$.) We also assumed (purely for convenience) that no individual entered the population between the start and end of the study (thus, since S<1, the estimated population size should decline over time). We also assumed no heterogeneity in capture probabilities among individuals. What about the γ parameters? We assumed a time-dependent Markovian model: $\gamma''_2 = 0.2$, $\gamma''_3 = 0.3$, $\gamma''_4 = 0.3$, $\gamma''_5 = 0.2$ and $\gamma'_3 = 0.2$, $\gamma'_4 = 0.4$, $\gamma'_5 = 0.3$.

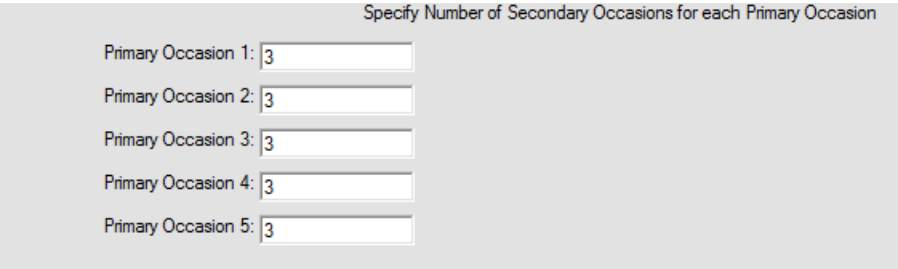
OK, now, let's analyze these simulated data in **MARK**. For our candidate model set, we'll assume that there are 3 competing models: (i) a model with no temporary emigration (i.e, $\gamma''_i = \gamma'_i = 0$), (ii) a model with random temporary emigration (i.e., $\gamma''_i = \gamma'_i$), and (iii) a model with Markovian temporary emigration (in this case, the 'true' model under which the data were simulated). We'll skip the 'even flow' model mentioned earlier for now. It is not a model we can build directly using PIMs. Moreover, building the 'even flow' DM requires a design matrix 'trick' we haven't seen before. For now, we're going to concentrate on simple model construction, using PIMs. We'll get back to the 'even flow' model later. In our analysis, we'll also assume we have 'prior knowledge' concerning the true structure for the encounter probabilities (i.e., the parameter structure for p_i and c_i). To facilitate referring to the models in the results browser, we'll call them simply '`no movement`', '`random movement`' and '`Markovian movement`', respectively.

Start **MARK** and select the '**`Robust Design`**' data type on the model specification window. **MARK** will immediately 'pop-up' a small sub-window, asking you specify the model type for the closed captures data type (recall that you're modeling encounters during secondary samples using a closed population estimator). For this example, we'll use '**`Full Likelihood p and c`**'. After selecting the appropriate input file (`rd_simple1.inp`), we need to tell **MARK** how many occasions we have. For the robust design, we need to do this in stages. First, how many total occasions? In this case, we have 5 primary occasions, each of which consists of 3 secondary occasions. So, 15 total occasions.

Now, the next stage is specifying the primary and secondary sampling structure. In other words, how are the secondary samples divided among primary samples? If you look at the `.INP` file, there is no obvious indication in the file itself where the break-points are between primary occasions. However, **MARK** has a useful feature which makes specifying the primary and secondary sample structure relatively straightforward. If you look immediately to the right of where you entered the total number of occasions, you'll see the usual '**`Set Time Intervals`**' button. Immediately above and to the right of the '**`Set Time Intervals`**' button is a button labeled '**`Easy Robust Design Times`**'. Why 2 buttons? Well, you could specify the primary and secondary sampling model structure by appropriately setting the time intervals (see below), or you can take the 'easy way out' (pun intended) by using the '**`Easy Robust Design Times`**' button. If you click this button, you're presented with a new window which asks you to specify the number of primary sampling occasions:



In our case, we have 5 primary sampling occasions. Once you click the '**`OK`**' button, **MARK** responds with a second pop-up window, asking you to specify the number of secondary sampling occasions for each primary session.



The default values that you will see are derived simply by taking the total number of occasions (15 in this example) and dividing that number by the number of primary sampling occasions (5) – in this example, the default of 3 secondary sampling occasions conveniently matches the true structure of our sampling – of course, if it didn't, then we would simply manually adjust the number of secondary sampling occasions per primary sampling occasion, subject to the constraint that the total number of secondary occasions (summed over all primary sampling occasions) equaled 15 (in our example).

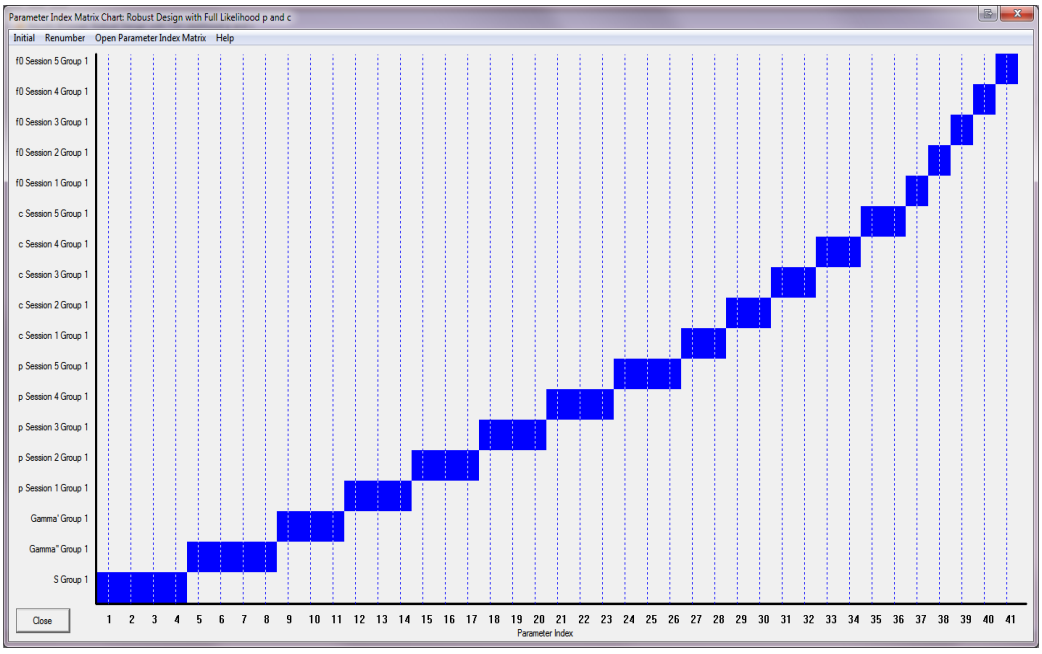
In fact, what the '**Easy Robust Design Times**' button is doing is setting the time intervals used to specify which encounter occasions are grouped together to form the secondary encounter sessions within each primary period. The time intervals between the encounter occasions within a primary session have a length of zero, whereas the time intervals between primary sessions have a positive (>0) length.

An example will make this clearer. Assume that animals are trapped for 15 separate times. The first year, animals are trapped for 2 days, the second year for 2 days, the third year for 4 days, the fourth year for 5 days, and the fifth year for 2 days. The number of encounter occasions would be specified as 15. The length of the time intervals would be specified as

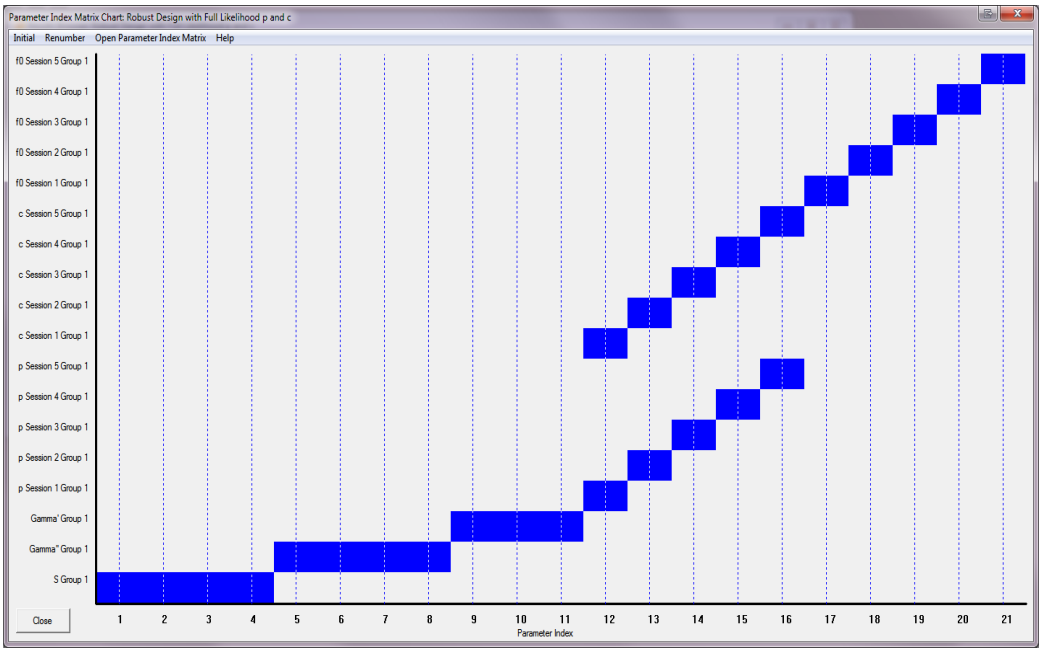
```
0  1  0  1  0  0  0  1  0  0  0  0  1  0
```

That is, only 14 time intervals are needed, where the value 1 means that 1 year elapsed. This mechanism is flexible, but can be a bit tricky (hence, the utility of the '**Easy Robust Design Times**' button). Note that all sessions must have at least 2 occasions. Thus, you will never have 2 consecutive time intervals of length > 0.

Once you have correctly specified the primary and secondary sampling structure (by whichever method you chose), click the '**OK**' button. As usual, **MARK** responds by presenting you with the PIM for the first parameter, in this case, the survival parameter S. Let's look at the PIM chart – but, remember how many parameters you're dealing with here: you have survival (S), the γ parameters (γ' and γ''), the encounter parameters (p and c), and the population size N (for simple closed capture models where N is a parameter in the likelihood – see Chapter 14). Meaning, the PIM chart will be very big (very...). Even for this simple example, with 'only' 15 total occasions, the PIM chart (shown at the top of the next page) is 'dense' with information (to put it mildly).

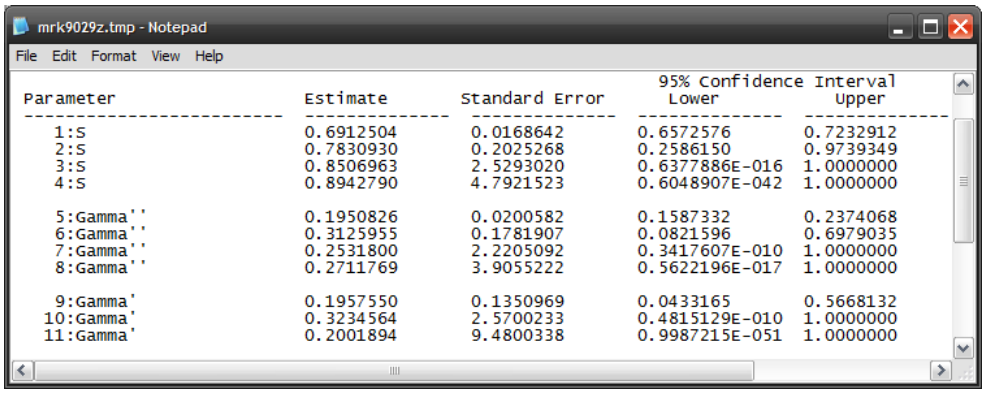


We'll start by trying to fit a model with time dependence in S, γ', and γ'', but where $p_i = c_i = c_.$ for each primary occasion i (although we allow annual p to vary). The PIM chart corresponding to this model is shown below:



If you've read the preceding text carefully, you'll recognize that in fact (i) this represents a Markovian model for the γ parameters, and (ii) without constraints, there will be identifiability problems for S and γ for this model.

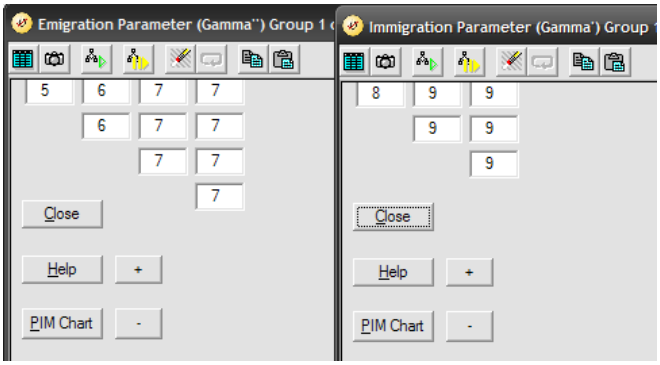
We can confirm this by running the model, and looking at the estimates for S and γ from this model:



Parameter	Estimate	Standard Error	95% Confidence Interval Lower	Upper
1:S	0.6912504	0.0168642	0.6572576	0.7232912
2:S	0.7830930	0.2025268	0.2586150	0.9739349
3:S	0.8506963	2.5293020	0.6377886E-016	1.0000000
4:S	0.8942790	4.7921523	0.6048907E-042	1.0000000
5:Gamma''	0.1950826	0.0200582	0.1587332	0.2374068
6:Gamma''	0.3125955	0.1781907	0.0821596	0.6979035
7:Gamma''	0.2531800	2.2205092	0.3417607E-010	1.0000000
8:Gamma''	0.2711769	3.9055222	0.5622196E-017	1.0000000
9:Gamma'	0.1957550	0.1350969	0.0433165	0.5668132
10:Gamma'	0.3234564	2.5700233	0.4815129E-010	1.0000000
11:Gamma'	0.2001894	9.4800338	0.9987215E-051	1.0000000

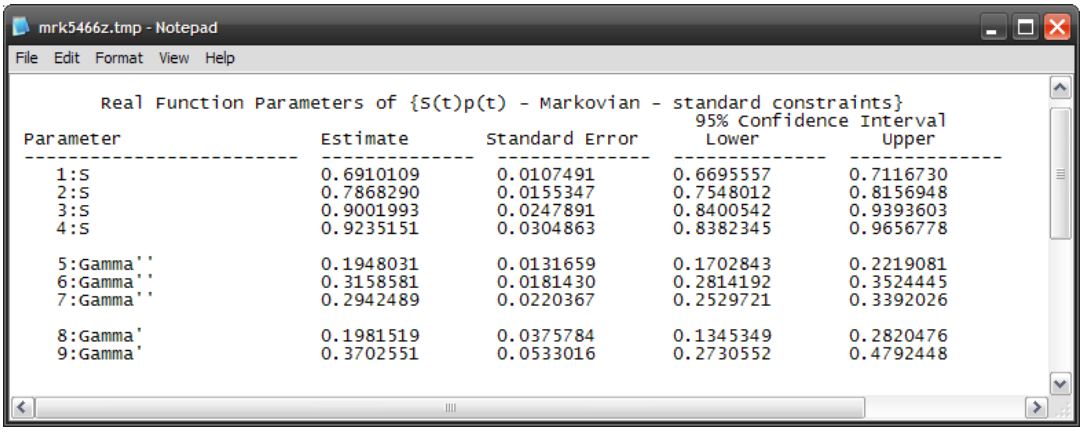
We see that the estimates for the last two S and γ parameters are completely confounded. Now, let's see what happens to the estimates if we apply the constraints $\gamma'_k = \gamma'_{k-1}$, and $\gamma''_k = \gamma''_{k-1}$? As mentioned earlier, these constraints are necessary to make S and the remaining γ parameters identifiable. How do we set these constraints?

Here, we'll use a simple PIM-based approach. Here are the modified PIMs for γ'' and γ', respectively:



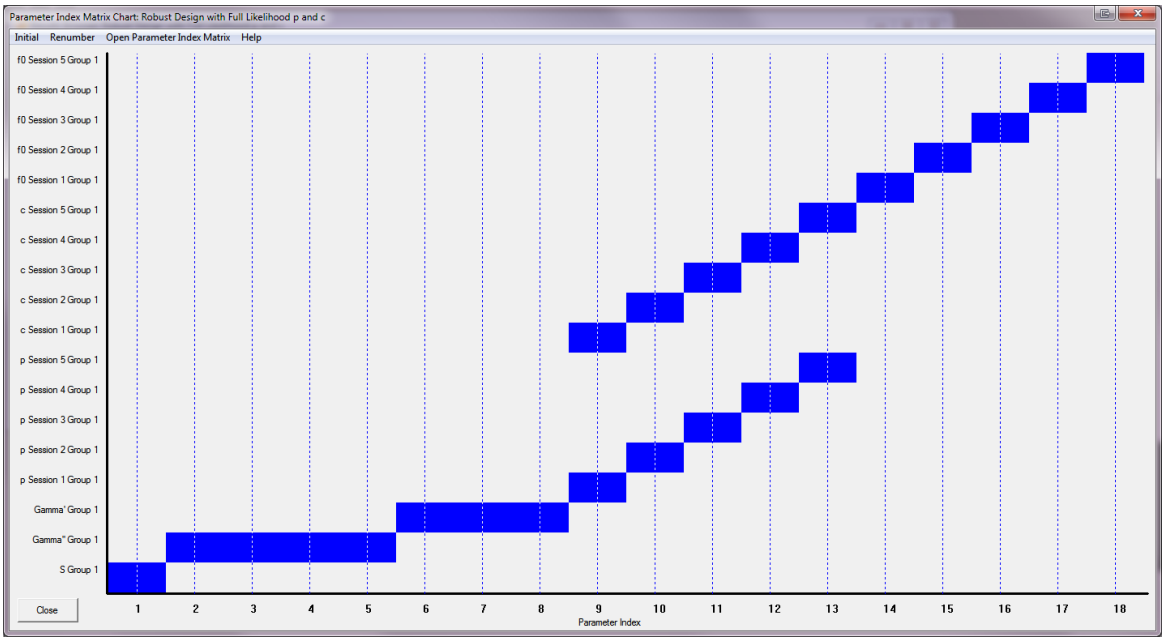
Make sure you understand what we've done in the PIMs. We've set the last two parameters equal to each other for both γ'' (parameter index 7) and γ' (parameter index 9). This constraint should allow us to estimate γ''_2 (parameter index 5) and γ''_3 (parameter index 6), and γ'_3 (parameter index 7).

If we fit this model to the data, we see that the estimates of S and γ (shown at the top of the next page) are all reasonable, and quite close to the true underlying parameter values used in the simulation ($S_1 = 0.7, S_2 = 0.8, S_3 = 0.9, S_4 = 0.8$; $\gamma''_2 = 0.2, \gamma''_3 = 0.3; \gamma''_4 = 0.3, \gamma''_5 = 0.2$ and $\gamma'_3 = 0.2$). Remember that we've achieved this 'identifiability' by applying constraints on the terminal pairs of γ parameters – not only may there be no good biological justification for imposing this constraint, but the estimates of the constrained γ (parameter index 7, representing the constraint $\gamma''_4 = \gamma''_5$, and parameter index 9, representing the constraint $\gamma'_4 = \gamma'_5$) are not biologically interpretable.

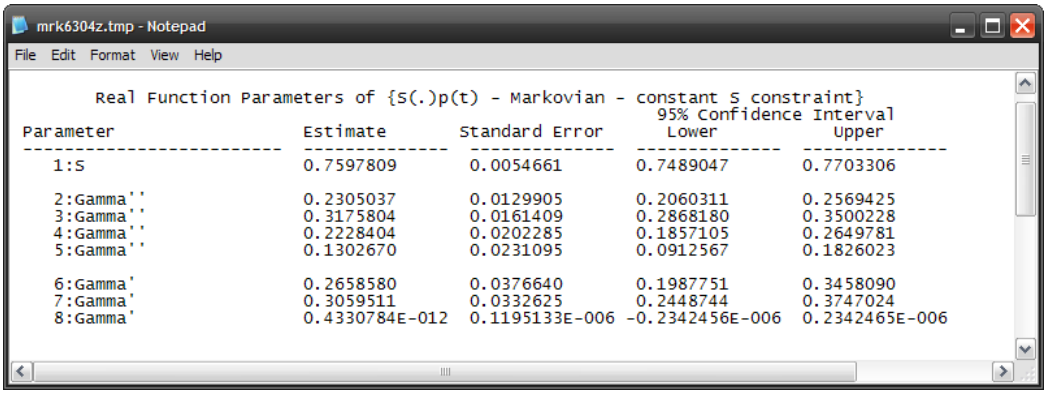


What if instead of constraining γ we'd applied a constraint to the survival parameter S? For example, what if we constrained S to be constant over time? As you'll recall from earlier chapters, constraining one parameter can often eliminate confounding with other parameters, and in the process, make them identifiable. For example, in a simple $\{\varphi_t p_t\}$ live mark-recapture model, the terminal φ and p parameters are confounded, whereas if you fit model $\{\varphi . p_t\}$ (i.e., constrain φ to be constant over time), all of the encounter probabilities p_t are estimable, including the terminal parameter. Of course, you would still want to have a good prior motivation to apply the constraint.

So, for our present analysis, what happens to our estimates if we (i) constrain S to be constant over time, and (ii) 'remove' the $\gamma'_k = \gamma'_{k-1}$ and $\gamma''_k = \gamma''_{k-1}$ constraints? Here is the PIM chart corresponding to this model – note that the indexing for γ'' is now $2 \rightarrow 5$ (whereas for the constrained model, it was $2 \rightarrow 4$), and for γ', the indexing is from $6 \rightarrow 8$ (instead of $6 \rightarrow 7$ for the constrained model).



Here are the estimates from fitting this model with constant survival to the data:

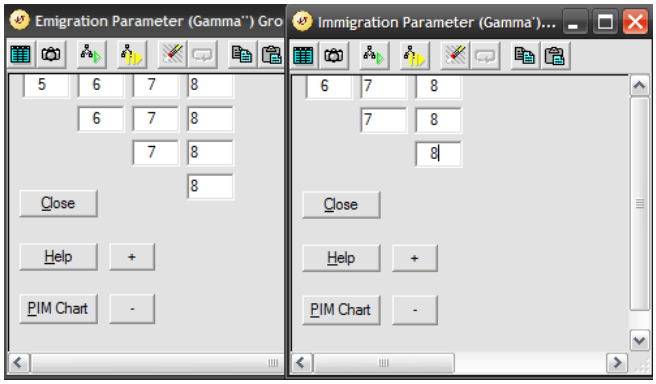


We see that in fact all of the γ'' parameters are now estimable, as are the first two estimates for γ'. The estimates qualitatively match the true underlying parameter values – differences reflect the fact that in the generating model used to simulate the data, survival S was time-dependent – here we are constraining it to be constant over time, which affects our estimates of other parameters.

Let's continue fitting the models in our candidate model set, assuming that S is time-dependent (go ahead and delete the model we just ran with S held constant from the browser – we ran that model just to demonstrate that you could achieve identifiability by hold S constant). We'll fit the '`Random movement`' model next. Recall that for a '`Random movement`' model, which is essentially the 'classical' robust design, we apply the constraint $\gamma''_i = \gamma'_i$.

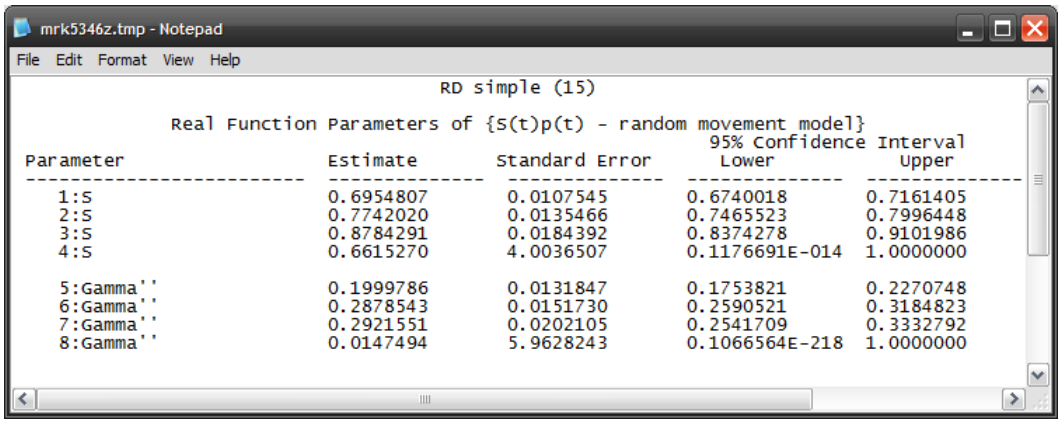
Also remember that without additional constraints, the parameters γ''_k, γ'_k, and S_{k-1} are all confounded. While you could set some constraints to 'pull them apart', in practice it is often easier to forgo the constraint – in that case, you would simply ignore the estimates of S_{k-1}, γ'_k, and γ''_k. Estimates of the remaining parameters would be unbiased.

Specifying the '`Random movement`' model is straightforward, but remember that there is one more γ'' than γ' parameter. In this case, there is no γ'_2 parameter corresponding with γ''_2, so we apply the constraint to the γ parameters for primary occasions 3 and 4 only. Again, this is most easily accomplished by modifying the PIMs for γ'' and γ', respectively:

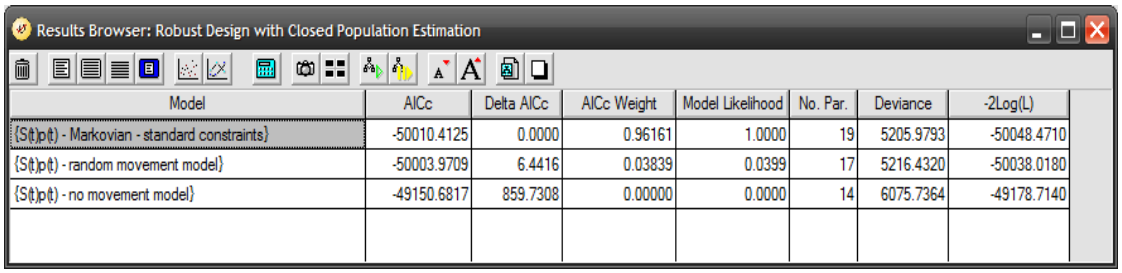


Run this model and add the results to the browser.

If you look at the real estimates from the 'Random movement' model (shown below) you see clearly that the final S and γ parameters are confounded.



Finally, the 'No movement' model. Recall that to fit this model, we fix $\gamma_i' = 1$ and $\gamma_i'' = 0$ over all occasions. We can do this easily by using the '**Fix parameters**' button in the '**Run numerical estimation**' window. Since we just finished building the 'Random movement' model, we can run the 'No movement' model simply by fixing all of the γ parameters in the 'Random movement' model (parameters 5 → 8) to either 1 or 0 (for γ' and γ'' respectively). Go ahead and fix the γ's, run the 'No movement' model, and add the results to the browser:



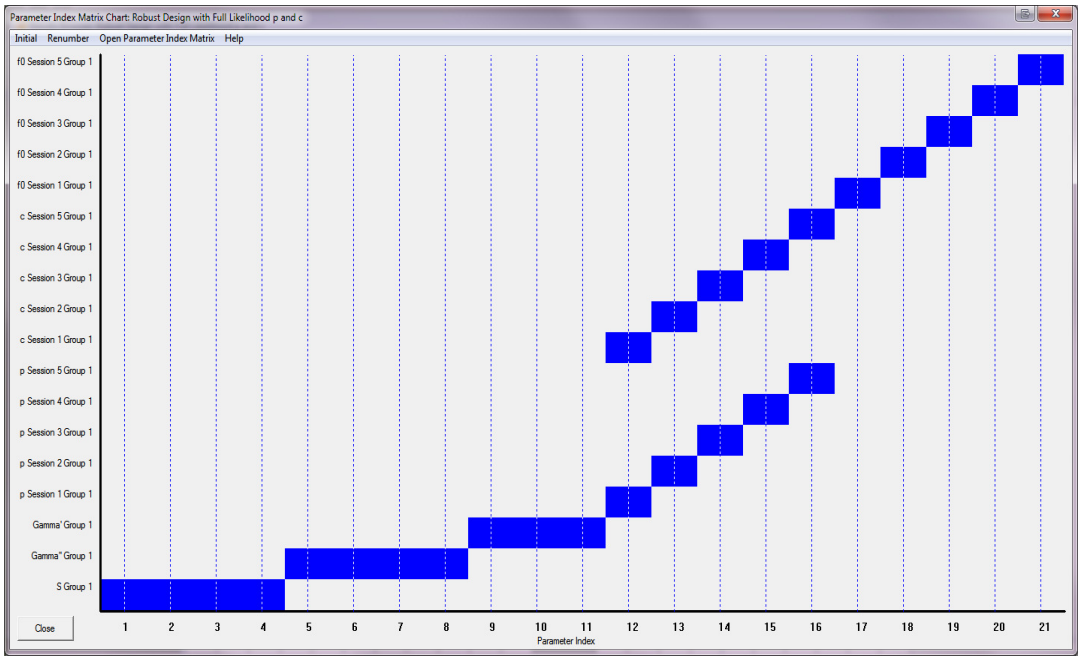
Model	AICc	Delta AICc	AICc Weight	Model Likelihood	No. Par.	Deviance	-2Log(L)
{S(t)p(t) - Markovian - standard constraints}	-50010.4125	0.0000	0.96161	1.0000	19	5205.9793	-50048.4710
{S(t)p(t) - random movement model}	-50003.9709	6.4416	0.03839	0.0399	17	5216.4320	-50038.0180
{S(t)p(t) - no movement model}	-49150.6817	859.7308	0.00000	0.0000	14	6075.7364	-49178.7140

Note that the reported AIC_c values are all negative. An explanation for negative AIC_c values was presented in Chapter 14. Negative AIC_c values are legitimate and interpreted in the same way as positive AIC_c values. Keep in mind that minimum AIC_c remains the target and the model with the 'most negative' AIC_c, i.e., the one furthest from zero, is the most parsimonious model.

In looking at our results, we conclude that the 'Markovian model' has by far the most support in the data among our 3 candidate models. This should not be surprising – a Markovian model was the generating model for the simulated data.

Using the design matrix in the RD – simple example revisited...

In the preceding, we built the models using PIMs. How would we build these models using the design matrix (DM)? We start by considering the PIM structure for a model with full time-dependence in S and γ, with annual variation in p (in fact, this is the generating model used to simulate the data we considered in the preceding example). The PIM chart corresponding to this structure is shown at the top of the next page.



There are a number of ways to build the DM corresponding to this PIM chart. One way, which is arguably the 'default' approach, is shown in the following.

B1:	B2:	B3:	B4:	B5:	B6:	B7:	B8:	B9:	B10:	B11:	Parm	B12:	B13:	B14:	B15:	B16:	B17:	B18:	B19:	B20:	B21:
1	1	0	0	0	0	0	0	0	0	0	1:S	0	0	0	0	0	0	0	0	0	0
1	0	1	0	0	0	0	0	0	0	0	2:S	0	0	0	0	0	0	0	0	0	0
1	0	0	1	0	0	0	0	0	0	0	3:S	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	4:S	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	1	0	0	0	0	0	5:Gamma"	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	1	0	0	0	0	6:Gamma"	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	1	0	0	0	7:Gamma"	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0	0	0	8:Gamma"	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	1	1	0	9:Gamma'	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	1	0	1	10:Gamma'	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	1	0	0	11:Gamma'	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	12:p Session 1	1	1	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	13:p Session 2	1	0	1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	14:p Session 3	1	0	0	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	15:p Session 4	1	0	0	0	1	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	16:p Session 5	1	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	17:f0 Session 1	0	0	0	0	0	1	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	18:f0 Session 2	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	0	0	0	0	0	19:f0 Session 3	0	0	0	0	0	0	0	1	0	0
0	0	0	0	0	0	0	0	0	0	0	20:f0 Session 4	0	0	0	0	0	0	0	0	1	0
0	0	0	0	0	0	0	0	0	0	0	21:f0 Session 5	0	0	0	0	0	0	0	0	0	1

Here, we assume that we're going to model each of the structural parameters (S, γ', γ'', p) independently of each other. Meaning, each parameter will have its own intercept. While there is nothing wrong with this, it makes it somewhat more difficult to build models where we want (or need) to specify particular relationships between 2 or more of the parameters. For example, there is no simple

modification of this DM which will let you build a '`Random movement`' model, where $\gamma' = \gamma''$.

Is there a more flexible approach? In fact, you might recall from our development of the DM for closed population abundance models (Chapter 14 – section 14.6) that a straightforward approach is to consider each of the parameters you want to model 'together' (i.e., as being related to each other in some fashion) as different levels of a putative 'group' factor, using a common intercept for these parameters. [In fact, you may recall that we first introduced this concept back in Chapter 7 with respect to 'age' and 'time since marking' models]. We start by specifying a putative parameter 'group' for the γ parameters – we'll call it '`gg`' (for 'γ-group'). Next, to help us keep track of what we're doing, we write out the linear model corresponding to the γ parameters in the PIM chart shown on the preceding page. We know that without constraints not all parameters are identifiable, but it represents our most general parametrization for γ, which we will constrain to build our 3 candidate models. Here is the linear model corresponding to the γ parameters shown in the PIM chart:

$$\begin{aligned} '\gamma' &= \texttt{INTCPT+gg+TIME+gg.TIME} \\ &= \beta_1 + \beta_2(\texttt{GG}) + \beta_3(\texttt{T}_1) + \beta_4(\texttt{T}_2) + \beta_5(\texttt{T}_3) + \beta_6(\texttt{gg.T}_2) + \beta_7(\texttt{gg.T}_3) \end{aligned}$$

We see there are 7 terms in the linear model, which correctly matches the 7 parameters for γ specified in the PIM chart. Note that we model only 'plausible interactions'. Since there is no γ'_1, parameter, then no interaction is specified for interval coded by $\texttt{T}_1$.

Here is what this linear model for γ would look like coded in the DM:

4:S	0	0	0	0	0	0	0	0
5:Gamma''	1	1	1	0	0	0	0	0
6:Gamma''	1	1	0	1	0	1	0	0
7:Gamma''	1	1	0	0	1	0	1	0
8:Gamma''	1	1	0	0	0	0	0	0
9:Gamma'	1	0	0	1	0	0	0	0
10:Gamma'	1	0	0	0	1	0	0	0
11:Gamma'	1	0	0	0	0	0	0	0

(Note that we'll leave the coding for the other parameters S, p and N the same).

Given this DM, how would we modify it to construct the 3 models we constructed in the preceding section using the PIM approach? We'll start with the '`Markovian model`'. Recall for a model where movement is modeled as Markovian, we generally need to set (i) $\gamma''_k = \gamma''_{k-1}$ and (ii) $\gamma'_k = \gamma'_{k-1}$. How would we set these constraints in the DM? The key is in realizing that by setting $\gamma''_k = \gamma''_{k-1}$ and $\gamma'_k = \gamma'_{k-1}$ we are, in effect, 'merging' the final two time intervals. In other words, instead of having 4 time intervals, we now have 3. Now, have another look at the part of the DM relating to the γ parameters (shown above). Note that column `B9` (labeled `t3`) specifies the 3rd time interval. So, to make time interval 3 equivalent to time interval 4 (which we coded by convention as the reference interval), we simply delete column `B9`. But, if we delete the `t3` column (i.e., `B9`), when we also need to delete any interaction column involving `t3`. So, we must also delete column `B11` (labeled `gg.t3`).

Here is what the modified DM for the '`Markovian model`' for γ looks like:

Parm	B5: int - gamma	B6: gg	B7: t1	B8: t2	B9: gg t2
1:S	0	0	0	0	0
2:S	0	0	0	0	0
3:S	0	0	0	0	0
4:S	0	0	0	0	0
5:Gamma"	1	1	1	0	0
6:Gamma"	1	1	0	1	1
7:Gamma"	1	1	0	0	0
8:Gamma"	1	1	0	0	0
9:Gamma'	1	0	0	1	0
10:Gamma'	1	0	0	0	0
11:Gamma'	1	0	0	0	0

If you run this model, and add the results to the browser – you'll see that they match the results from the '`Markovian model`' built using the PIM approach.

Next, we'll consider the '`random movement`' model. Recall for this model, we set $\gamma' = \gamma''$. This is analogous to setting $p = c$ in a closed population abundance model. By setting $\gamma' = \gamma''$, we are in effecting eliminating the difference between the groups. So, all we need to do is (i) delete the '`gg`' column (`B6`), and (ii) delete any interaction column involving '`gg`' (in this case, column `B9`, labeled '`gg.t2`'). Here is the modified DM for γ for the '`random movement`' model:

5:Gamma"	1	1	0	
6:Gamma"	1	0	1	
7:Gamma"	1	0	0	
8:Gamma"	1	0	0	
9:Gamma'	1	0	1	
10:Gamma'	1	0	0	
11:Gamma'	1	0	0	

Again, if you run this model, and add the results to the browser – you'll see that they match the results from the '`random movement`' model built using the PIM approach. Note that by simply modifying the DM we built for the '`Markovian model`', we are retaining the $\gamma''_k = \gamma''_{k-1}$ and $\gamma'_k = \gamma'_{k-1}$ constraints. Doing so (or not) does no affect the overall model fit, but does influence the identifiability of some parameters (in particular, the final estimate for survival S).

Finally, for the (null) '`no movement`' model, we want to set $\gamma' = 1$, and $\gamma'' = 0$. Starting from the DM we just specified for the '`random movement`' model, all we need to do is (i) delete the time columns (`B7` → `B9`), and (ii) fix parameters 5 → 8 to 0, and parameters 9 → 11 to 1 when we run the model. Results from this model should match those from the '`no movement`' model built using PIMs.

15.6.2. Closed robust design – more complex worked example

In the preceding, we used PIMs, and a PIM chart, to construct the various models in our candidate model set. We also demonstrated (and reinforced) the notion of parameter constraints that are often necessary

to make various parameters estimable. Recall in particular that for a model with Markovian movement, setting the constraints $\gamma''_{k-1} = \gamma''_k$ and $\gamma'_{k-1} = \gamma'_k$ (where k is the number of primary sampling occasions) makes the resulting 3 parameters estimable. However, as you might imagine, there is another approach which is often preferred, since it makes various parameters estimable, while avoiding constraints which work, but may have little biological meaning, or justification. This approach involves constraining various estimates to be functions of one or more covariates.

Consider the following scenario – suppose that only individuals in the breeding condition are available for encounter (this is quite plausible – for many taxa, only reproductively active individuals are ever encountered. Non-breeding individuals often do not return to the breeding site, and are thus not available for encounter). Suppose that in general, for some species, if the climatic conditions are favorable, the tendency of all individuals to breed is increased, relative to a year with harsher climatic conditions, where more individuals opt out of breeding. Thus, we would anticipate that in general in a 'good' year, $(1 - \gamma')$ and $(1 - \gamma'')$ will generally be greater than γ' and γ''. The reverse would generally be true in a 'poor' year.

For this example, we simulated 15 occasions worth of data – 5 primary sampling occasions, each with 3 secondary samples. Primary samples 1, 2 and 4 were classified as 'good' years, whereas primary samples 3 and 5 were taken in 'poor' years. In 'good' years, $\gamma'_g = 0.5$ and $\gamma''_g = 0.1$. In 'poor' years, $\gamma'_p = 0.7$ and $\gamma''_p = 0.25$. These values were chosen to reflect the basic expectation that in 'good' years, individuals that were breeders the previous year tend to remain in breeding state, whereas individuals that were not breeding the year before tend to become breeders. The reverse is likely true (in many cases) in 'poor' years. We also assumed that survival is marginally lower in 'poor' years than in 'good' years ($S_g = 0.8 > S_p = 0.7$). Finally, we also assumed that encounter effort tends to be lower in 'poor' years (perhaps for some logistical reasons related to the poorer weather), but that $p^*_i = c_i$ in all years. We set $p^*_g = c_g = 0.5$, and $p^*_p = c_p = 0.3$. We simulated a data set with 2,000 individuals captured, marked and released alive on the first occasion. We assumed closure within each primary sampling period, and no net immigration of new individuals into the population on any subsequent occasion (thus, expected population size N should decline over time). The simulated data are contained in the file `rd_complex1.inp`.

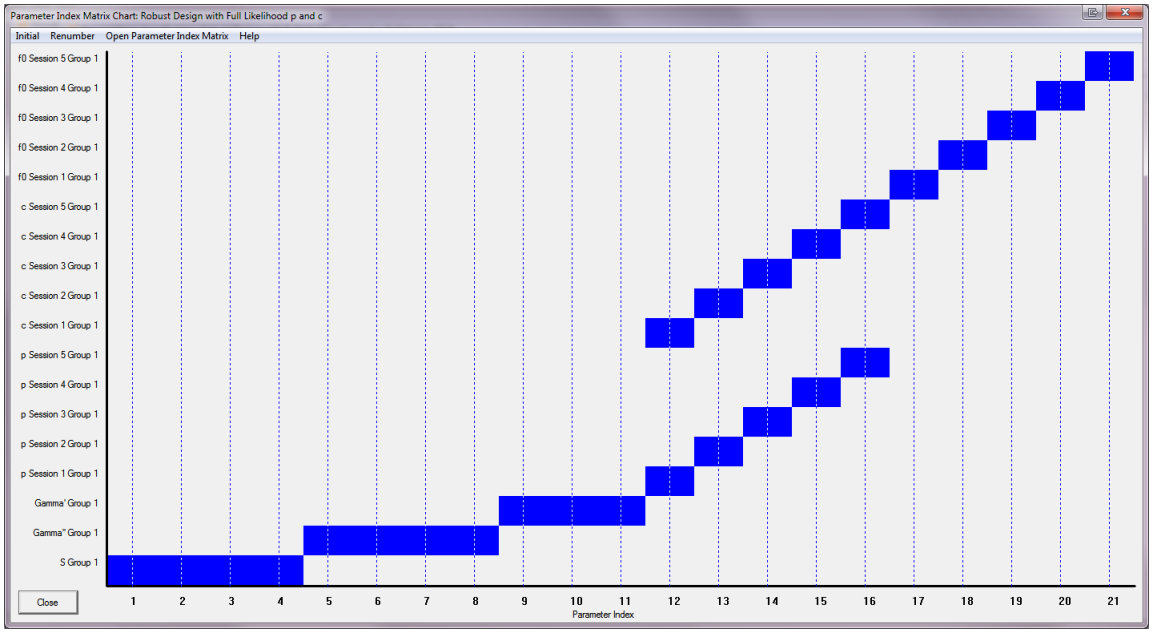
Now, let's proceed to analyze these data – with the intent of demonstrating that use of covariates can make it possible to estimate various parameters without relying on equality constraints as described earlier. We'll assume that our model set consists of 2 models:

1. Markovian, no covariates – simple time variation in S, γ, and $p = c$
2. Markovian, with covariates used to explain temporal variation in S, γ, and $p = c$.

Now, in this example, the covariate is a dichotomous indicator variable ('good' year or 'poor' year). As such, this example problem is analogous to the (by now) familiar European dipper 'flood' analysis. You may recall from Chapter 6 that there are two ways to approach this type of analysis. We could, in fact, use a PIM-only approach for some models, by coding 'good' and 'poor' directly into the PIMs for various parameters (see section 6.7 in Chapter 6). However, we also recall that ultimately this limits the types of models we want to build – more generally, we'd like to use a design matrix approach, since it will (ultimately) give us complete flexibility over the types of models we build. So, that is the approach we'll employ here for our model with covariates for 'good' and 'poor' years.

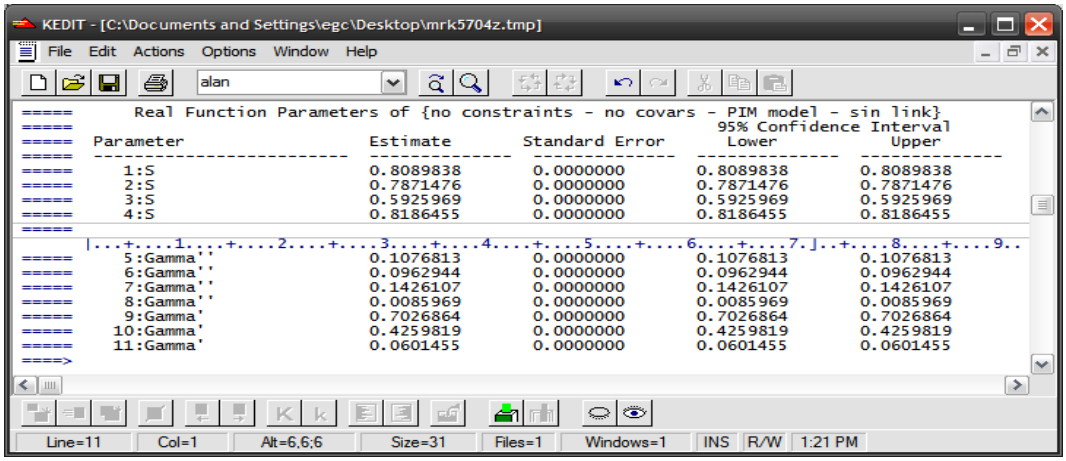
Start **MARK**, and access the `rd_complex1.inp` file. Select '**`robust design, closed captures`**' as the model type – 15 total occasions, with 5 primary occasions each consisting of 3 secondary samples. To start, we'll build the unconstrained model – time variation in S, γ, and $p = c$. We can do this most easily by making use of the PIM chart.

Go ahead and bring up the PIM chart, and modify it so it looks like the following:



This PIM chart is similar to the PIM chart used in the preceding (simpler) example – the only major difference is that now the 'blue boxes' for S, γ'' and γ' are all time-dependent. Recall that this has implications for estimability of several parameters.

Now, we recall from earlier sections of this chapter that for time-dependent robust design models, not all parameters are estimable without some constraints. Specifically, we would need to set the constraints $\gamma''_4 = \gamma''_5$ and $\gamma'_4 = \gamma'_5$ (in a few moments, we'll also discuss whether or not these particular constraints actually 'make sense' given the nature of 'this study'). However, for purposes of demonstrating the necessity of these constraints, let's first run the model without imposing the constraints. We see from the following listing of parameter estimates that indeed, none of the survival or γ parameters are estimable (they all have completely unrealistic SE or 95% CI).



Now, let's re-run this same model, after applying the constraints $\gamma''_4 = \gamma''_5$ and $\gamma'_4 = \gamma'_5$. In the preceding example, we applied these constraints by directly modifying the PIMs for both γ parameters. However, this is not necessary – we can apply these constraints using the design matrix, which we want to build anyway, for the purposes of constraining our estimates to be functions of 'good' or 'poor' years.

First, let's build the design matrix (DM) which corresponds exactly to the PIM chart shown on the preceding page – since our models set consists of only 2 candidate models (Markovian with and without covariates), we'll use the 'default' coding which treats γ' and γ'' separately (i.e., each parameter has its own intercept).

B1:	B2:	B3:	B4:	B5:	B6:	B7:	B8:	B9:	B10:	B11:	Parm	B12:	B13:	B14:	B15:	B16:	B17:	B18:	B19:	B20:	B21:
1	1	0	0	0	0	0	0	0	0	0	1:S	0	0	0	0	0	0	0	0	0	0
1	0	1	0	0	0	0	0	0	0	0	2:S	0	0	0	0	0	0	0	0	0	0
1	0	0	1	0	0	0	0	0	0	0	3:S	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	4:S	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	1	0	0	0	0	0	5:Gamma"	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	1	0	0	0	0	6:Gamma"	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	1	0	0	0	7:Gamma"	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0	0	0	8:Gamma"	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	1	1	0	9:Gamma'	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	1	0	1	10:Gamma'	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	1	0	0	11:Gamma'	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	12:p Session 1	1	1	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	13:p Session 2	1	0	1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	14:p Session 3	1	0	0	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	15:p Session 4	1	0	0	0	1	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	16:p Session 5	1	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	17:f0 Session 1	0	0	0	0	0	1	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	18:f0 Session 2	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	0	0	0	0	0	19:f0 Session 3	0	0	0	0	0	0	0	1	0	0
0	0	0	0	0	0	0	0	0	0	0	20:f0 Session 4	0	0	0	0	0	0	0	0	1	0
0	0	0	0	0	0	0	0	0	0	0	21:f0 Session 5	0	0	0	0	0	0	0	0	0	1

The basic coding should be familiar – however, since we're not modeling N as a function of anything, we specify simple temporal variation using an identity matrix for the part of the DM corresponding to N (lower right-hand corner).

Now, we want to modify this starting DM to constrain $\gamma''_4 = \gamma''_5$ and $\gamma'_4 = \gamma'_5$. Recall from the first, simpler example we worked through in this chapter that at present, the γ parameters for occasion k and $k-1$ are coded as separate time intervals. Thus, to apply the necessary constraint, we simply modify the DM so that the final two time intervals are treated as a single time step.

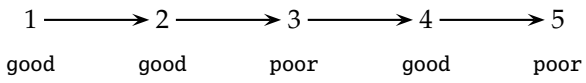
5:Gamma"	1	1	0	0	0	0
6:Gamma"	1	0	1	0	0	0
7:Gamma"	1	0	0	0	0	0
8:Gamma"	1	0	0	0	0	0
9:Gamma'	0	0	0	1	1	0
10:Gamma'	0	0	0	1	0	0
11:Gamma'	0	0	0	1	0	0
12:p Session 1	0	0	0	0	0	1

If you run this model, you'll see that now, all of the non-constrained γ parameters, and S are estimable (i.e., have reasonable standard errors, and are qualitatively close to the true parameter values). We can confirm this by modifying the PIMs directly (as in the preceding example). You should get precisely the same estimates as you just did using the design matrix. You will also note that the deviance of this

'constrained' model is identical to that for the unconstrained model we fit on the previous page (since the point estimates for the parameters are identical – all that has changed are the SE's).

But, again, we achieved 'estimability' at the cost of imposing some necessary, but perhaps not particularly 'biologically meaningful' constraints. Remember that for this example, primary samples 1, 2 and 4 were classified as 'good' years, whereas primary samples 3 and 5 were taken in 'poor' years. In 'good' years, $\gamma'_g = 0.5$ and $\gamma''_g = 0.1$. In 'poor' years, $\gamma'_p = 0.7$ and $\gamma''_p = 0.25$. Given this structure, it makes little *biological* sense to constrain $\gamma''_4 = \gamma''_5$ and $\gamma'_4 = \gamma'_5$. Although these constraints did make the non-constrained γ parameters estimable, there would be reason to be concerned about possible bias in the unconstrained estimates (relative to the true values).

It would seem to be more appropriate to modify the DM to account for variation between 'good' and 'poor' years. Let '1' be the dummy variable we use to code for a 'good' year, and '0' be the dummy variable coding for a 'poor' year. Recall that primary sessions 1, 2 and 4 occurred in 'good' years, while primary sessions 3 and 5 occurred in 'poor' years. Recall that we assume that S, γ, and $p = c$ are all functions of whether or not a year was classified as 'good' or 'poor'. Start by retrieving the model constructed using a DM *without* the logical constraints $\gamma''_4 = \gamma''_5$ and $\gamma'_4 = \gamma'_5$. We'll begin by modifying the coding for the survival parameter S first. In order to do this, we need to decide on whether or not S or γ over the interval from (i) to $(i+1)$ are functions of whether or not the environment is 'good' or 'poor' at time (i). For this example, we'll assume that since



that the estimate for parameter θ_i is a function of the state of the environment at the start of the interval from (i) to $(i+1)$. So, for example, S_1, S_2, and S_4 reflect 'good' years, whereas S_3 reflects a 'poor' year. In contrast, for parameters estimated *at* a particular sampling occasion (e.g., c, p, N), the estimate for θ_i reflects the conditions at sampling occasion i.

We'll start by modifying the part of the DM corresponding to survival.

Design Matrix Specification (B = Beta)

B0	B1	B2	B3	B4	B5	B6	B7	B8	Parm	E
1	1	0	0	0	0	0	0	0	1:S	0
1	1	0	0	0	0	0	0	0	2:S	0
1	0	0	0	0	0	0	0	0	3:S	0
1	1	0	0	0	0	0	0	0	4:S	0

The first column (labeled B0) is the intercept, while the second column (labeled B1) is the coding for 'good' or 'poor' years. Again, note that in our coding we are assuming that the conditions ('good' or 'poor') at primary occasion (i) determine the probability of surviving from occasion $(i) \rightarrow (i+1)$.

Now, what about γ' and γ''? Here, all we need to remember is that there is one fewer occasion for the γ' parameter (for reasons discussed earlier in this chapter). The primary challenge, then, is to keep track of which row refers to which occasion, for each of the two γ parameters. For $\gamma''_2 \rightarrow \gamma''_5$, we have 'good', 'good', 'poor' and 'good', whereas for $\gamma'_3 \rightarrow \gamma'_5$ we have 'good', 'poor' and 'good'.

Here is the completed DM for the γ parameters:

1	1	0	0	0	0	5:Gamma"	0
1	1	0	0	0	0	6:Gamma"	0
1	0	0	0	0	0	7:Gamma"	0
1	1	0	0	0	0	8:Gamma"	0
0	0	1	1	0	0	9:Gamma'	0
0	0	1	0	0	0	10:Gamma'	0
0	0	1	1	0	0	11:Gamma'	0

Finally, we modify the DM for the $p = c$ parameters (the DM structure for the population abundance parameter N is not changed). Remember that we're assuming that $p_i = c_i$ is equal to the conditions at sampling occasion (i). Thus,

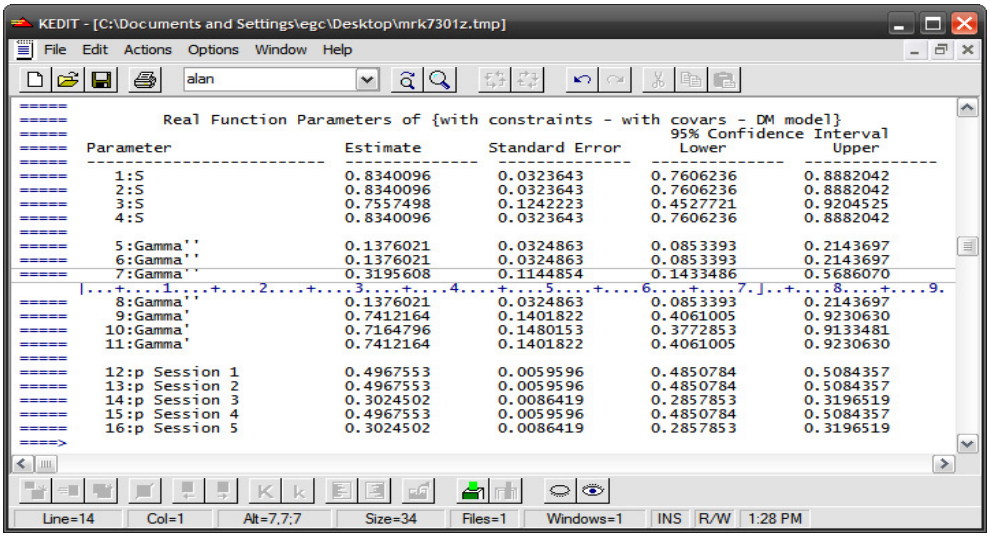
12:p Session 1	1	1	0	0	0	0	0
13:p Session 2	1	1	0	0	0	0	0
14:p Session 3	1	0	0	0	0	0	0
15:p Session 4	1	1	0	0	0	0	0
16:p Session 5	1	0	0	0	0	0	0
17:f0 Session 1	0	0	1	0	0	0	0
18:f0 Session 2	0	0	0	1	0	0	0
19:f0 Session 3	0	0	0	0	1	0	0
20:f0 Session 4	0	0	0	0	0	1	0
21:f0 Session 5	0	0	0	0	0	0	1

Not surprisingly, when we run this model and add the results to the browser (shown below), we see it has most of the support in the data (this is a good thing, since this is the 'true' model under which the data were simulated in the first place).

Results Browser: Robust Design with Closed Population Estimation

Model	AICc	Delta AICc	AICc Weight	Model Likelihood	No. Par.	Deviance
{with constraints - with covars - DM model}	-29575.1777	0.0000	0.52030	1.0000	13	3166.7940
{with constraints - no covars - DM model}	-29573.4582	1.7195	0.22023	0.4233	18	3158.4715
{no constraints - no covars - PIM model - sin link}	-29571.5890	3.5887	0.08649	0.1662	19	3158.3297
{no constraints - no covars - PIM model - logit link}	-29571.5890	3.5887	0.08649	0.1662	19	3158.3297
{no constraints - no covars - DM model}	-29571.5890	3.5887	0.08649	0.1662	19	3158.3297

However, what is of greater interest here is the influence of adding covariates to the DM on the estimability of the various parameters in the model. As shown at the top of the next page all of the parameters are estimable (dichotomized between 'good' and 'poor' years). It is worth noting that we assume the survival process is the same for those that are or are not available at any given time. We cannot derive a separate estimate of survival for individuals in and outside of the sample – to do so requires different approaches, discussed elsewhere.



15.7. The multi-state closed RD

In section 15.3.1 we briefly described the analogy between a model including temporary emigration from a single study area and a multi-state model with two states. I will illustrate this in more detail in this section, for two reasons. First, the multi-state closed robust design (hereafter, MSCRD) model, initially presented by Nichols and Coffman (1999), provides much more flexibility than either the original robust design model (which only permits two states, one of which must be unobservable) or the multi-state model (which lacks the extra information available from multiple secondary capture periods). However, as mentioned earlier, with flexibility comes complexity, and you will see that setting up the multi-state robust design model in **MARK** can be very involved (and tedious). Second, this section provides a segue to the multi-state open robust design (MSORD). In **MARK**, for the open robust design there is no simpler alternative to the full multi-state version. Again, the flexibility of this model will compensate for the complexity.

Between Chapter 10 and this chapter up to this point, the pieces of the MSCRD have already been explained individually. For someone familiar with the MS model of Chapter 10, the simplest way to view the MSCRD model is to note that each time capture probability p_i^s for state s appears in a MS model, it is replaced with p_i^{*s}. As in previous sections, if there are three secondary capture periods for primary period i, then the effective capture probability for state s in primary capture period i might be $p_i^{*s} = 1 - (1 - p_{i1}^s)(1 - p_{i2}^s)(1 - p_{i3}^s)$. The easiest way to illustrate the relationship between the classical robust design model and the MSCRD model is through an example. For simplicity we'll use the simple robust design example from section 15.6.1.

15.7.1. multi-state closed RD – simple worked example

The simple example discussed in section 15.6.1 involved a hypothetical study consisting of 4 primary periods of interest. Capture effort for each of these primary periods consisted of 3 secondary capture periods, conducted over a sufficiently short period of time that it is reasonable to assume the group of animals in the study area did not change (no deaths or births or movement in or out). This constitutes a robust design. The data were generated from a population with time-varying survival ($S_1 = 0.7, S_2 = 0.8, S_3 = 0.9, S_4 = 0.8$). Capture probability varied across primary periods ($p_{1j} = 0.5, p_{2j} = 0.6, p_{3j} = 0.6,$

$p_{4j} = 0.5$,$p_{5j} = 0.7$), but not within primary period, and recapture probability within a primary period was the same as initial capture probability (i.e., no trap effect; model M_0 – see Chapter 14). All of these parameters are also found in the MSCRD model. Notation changes when we consider the transition (in this case movement) parameters (i.e., γ and ψ). Relating the two sets of notation to the parameter values chosen, recall from p. 7 (in this chapter) that

$$\begin{aligned} \gamma' &\equiv \psi^{UU} \\ 1-\gamma' &\equiv \psi^{UO} \\ \gamma'' &\equiv \psi^{OU} \\ 1-\gamma'' &\equiv \psi^{OO} \end{aligned}$$

where where superscript '*O*' means 'observable', or *within the study area*, and superscript '*U*' means 'unobservable', or *outside the study area*.

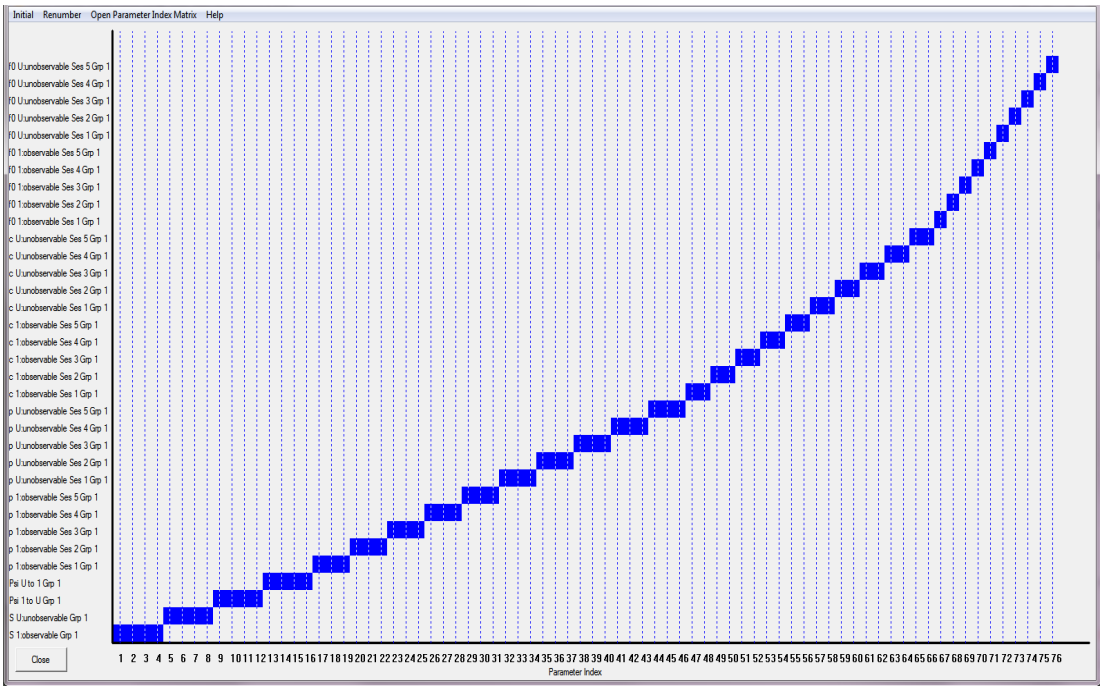
In this example $\psi_1^{OU} \equiv \gamma_2'' = 0.2$, $\psi_2^{OU} \equiv \gamma_3'' = \psi_3^{OU} \equiv \gamma_4'' = 0.3$, $\psi_4^{OU} \equiv \gamma_5'' = 0.2$, and $\psi_2^{UU} \equiv \gamma_3' = 0.2$, $\psi_3^{UU} \equiv \gamma_4' = 0.4$, $\psi_4^{UU} \equiv \gamma_5' = 0.3$. The analogy between the classic closed RD and the MS equivalent will become even clearer after running this in **MARK** using the MSCRD model and comparing it to the output from the original 'classic' analysis.

For this exercise, start by making a copy of '`rd_simple1.inp`' and call it '`MS_rdsimple1.inp`'. To use the MSCRD model simply open the **MARK** screen for developing new models and click on '**`Closed Robust Design Multi-state`**' (almost the last model listed). For consistency with our previous run with these data, choose the '**`Full Likelihood p and c`**' option. As in the previous example, after selecting the input file, specify 15 encounter occasions (remember there were 5 primary periods, each consisting of 3 secondary capture occasions). Click on '**`Easy Robust Times`**' and specify 5 primary periods. It will give you a new screen that happens to have the correct allocation of capture occasions across primary periods (3 in each). If in doubt, check back to section 15.6.1 where some of these steps were first introduced.

Up to this point the process has been identical to using the '**`Robust Design`**' option in **MARK**. Now we run into the first difference, and complication. Because this is a *multi-state* model, you see that the '**`State`**' and '**`Enter State Names`**' sections are now active. From Chapter 10 recall that you must specify for each state the code that will be used in the capture history to denote capture in that state. You also have the option of specifying a label for that state that might be more descriptive than the code alone. The default value of 2 states is appropriate for this example, because we have two statesin this example (`0` = observable = available for capture in the study area, `U` = unobservable = outside the study area). Click on '**`Enter State Names`**' and you will see the default codes of '`A`' for state 1 and '`B`' for state 2. Given that we are using a copy of '`rd_simple1.inp`', which denoted capture with a '`1`', you should replace the '`A`' with a '`1`' (or start over and this time replace the '`1`'s' in the input file with '`0`' or some other code. What code should you use for the second state? We are calling it state `U` for 'unobservable', so you could use that code. However, it does not really matter, since by definition you never capture the animal while in that state.

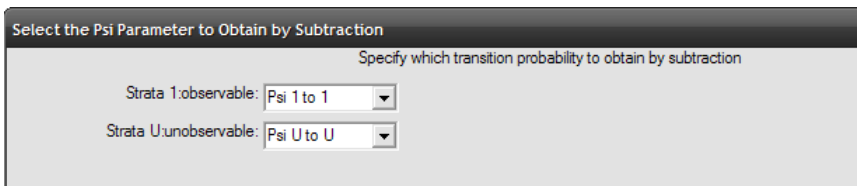
Caution: Be sure, however, not to use zero as the code, because zero is *always* reserved to denote non-capture.

When this is complete, and after you return to the main screen and click '**OK**', have a look at the PIM chart, shown below:



It is even more 'busy' than under the classic '**Robust Design**' model. If animals in each state could be captured, then the PIM chart could remain this complex. However, in this case of an unobservable state we will pare it down considerably. In addition, you can also reduce its size by opting for a 'Huggins' option when you first invoke a robust design model. As explained in Chapter 14, this designation denotes that abundance (N_i) is not a parameter of the model, but is derived after the fact from the model parameters and the number of animals captured. We will make three major changes in order to make this analysis identical to our analysis in section 15.6.1: (1) change the PIM definitions, (2) equate survival for both states, and (3) set capture probability to 0 for the unobservable state.

Changing the PIM definition would not be necessary for two of the three models we will run, but is necessary for one of them. It also permits us to equate γ values from the Robust Design option with ψ values from the MSCRD model. Recall from Chapter 10 that under the multi-state model, transition probabilities for a given state need to sum to 1.0, thus one of the probabilities is computed by subtraction from 1.0. As with the '**Multi-state Recaptures only**' option in **MARK**, the default is for the probability of remaining in a state to be gotten by subtraction. In the '**Robust Design**' option in **MARK** the γ parameters always refer to being outside the study area, so we will mimic that case here using the ψ parameters in the MSCRD model. First click on the '**PIM**' menu, then on '**Change PIM definitions**'. This should spawn a window looking like the one shown below.



For each state you can designate which transition is obtained by subtraction. For state `U` change '**`Psi U to U`**' to '**`Psi U to 1`**'. That way $\psi_i^{uu} = \gamma'_{i+1}$ will be estimated directly as a parameter.

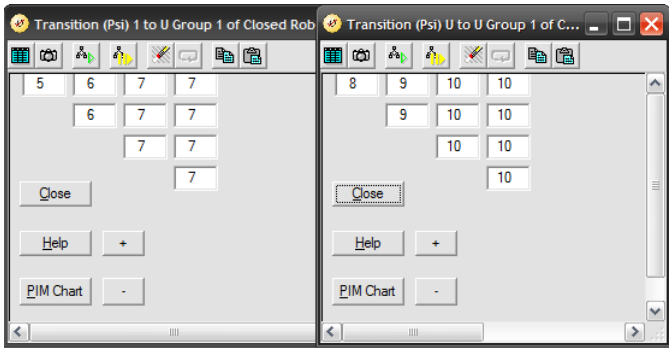
Next, you want to set survival equal for both states, which was one of the assumptions of the robust design model above. From previous chapters you will know that this can be done by dragging PIM's in the PIM chart, by copying one PIM to another after opening any PIM and then clicking on the '**`Initial`**' option at the top of the screen, or by opening a PIM and changing one entry at a time.

Finally, to account for the unobservable state you want to fix the capture and recapture probabilities for that state to 0. The simplest way to do it, especially when running many models, is to collapse all p and c PIM's for the unobservable state to one parameter. [Although for simplicity we don't do it here, to make it even easier to keep track of, designate that parameter you are going to fix to be parameter number 1 (move it to the far left of the PIM chart). In this way, you will always be fixing the same parameter to be 0. Otherwise, as you expand and contract the PIM chart with more restrictive or more general models, you will need to keep changing which parameter you fix to 0]. Recall that parameters are fixed by going to the '**`Run Current Model`**' screen and clicking on '**`Fix Parameters`**'.

To compare the MSCRD approach against the usual robust design model we will set up and run the same three models that we used in section 15.6.1: '`Markovian movement`', '`Random movement`', and '`No movement`'. The following is a reminder of the constraints needed for each of the models, for both the classic 'γ' RD parametrization, and the equivalent MS closed RD.

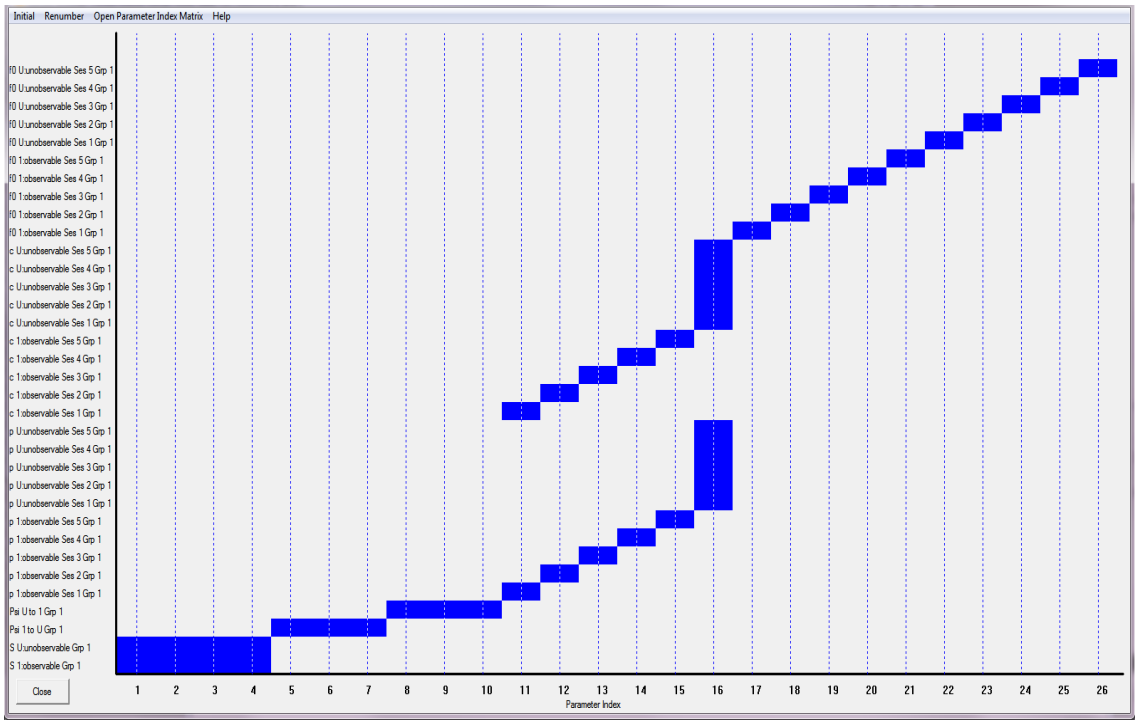
	γ formulation	*MS (ψ) formulation*
no movement	$\gamma' = 1, \gamma'' = 0$	$\psi^{uo} = \psi^{ou} = 0$
random movement	$\gamma' = \gamma''$	$\psi^{uu} = \psi^{ou}$
Markovian movement	$\gamma'_k = \gamma'_{k-1}$ $\gamma''_k = \gamma''_{k-1}$	$\psi_k^{oo} = \psi_{k-1}^{oo}$ $\psi_k^{uu} = \psi_{k-1}^{uu}$

We'll start with the '`Markovian movement`' model. For each type of movement we set the last two equal to one another (i.e., $\psi_3^{ou} = \psi_2^{ou}$, $\psi_3^{uu} = \psi_2^{uu}$), although this constraint is not necessary when survival is set equal over time. Recall that ψ_1^{uu} does not really enter the likelihood because no animals are released in the unobservable state. You can fix this parameter to any value or leave it alone. It will not affect the other parameters, but remember not to try to interpret it. The PIMs for the movement parameters are shown below:



All of the encounter probabilities are set constant within primary period, and '`c=p`' (corresponding to

model M_o) – however, recall that the unobservable state U is just that – unobservable, and thus encounter probabilities is 0 for all primary occasions. Finally, under the assumptions of the robust design, we set the survival probabilities for observed and unobserved states equal to each other. Here is the PIM chart:



Go ahead and run this model – remember to fix parameter 16 (i.e., the encounter probability for the unobservable state) first. You can also choose to fix ψ_1^{UU} (parameter 8 in the PIM chart) to some value (say, 0.5), or not – regardless, remember not to try to interpret it (for reasons noted above). We'll fix it to 0.5. The real parameter estimates of the survival S and movement parameters ψ are shown below. Note that they are virtually identical to the estimates for survival and γ from the Markovian model we fit using the classic closed RD (p. 19).

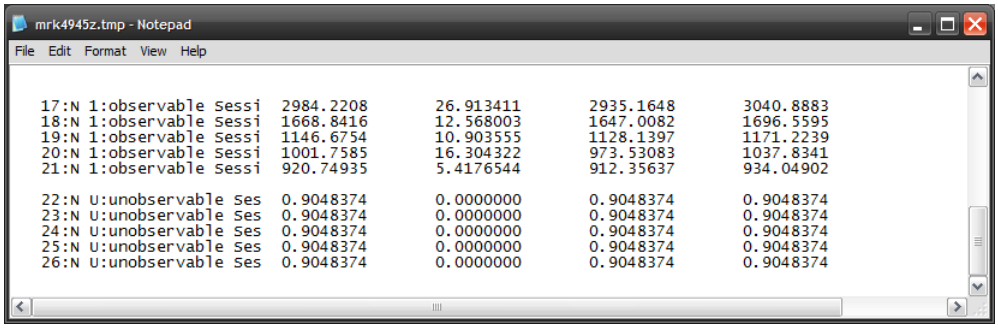
mrk6670z.tmp - Notepad

File Edit Format View Help

Real Function Parameters of {MSORD - Markovian}

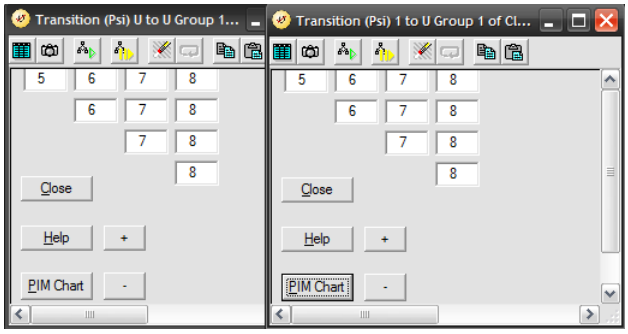
Parameter	Estimate	Standard Error	95% Confidence Interval Lower	Upper	
1:S 1:observable	0.6910116	0.0107492	0.6695563	0.7116738	
2:S 1:observable	0.7868300	0.0155351	0.7548013	0.8156965	
3:S 1:observable	0.9002063	0.0247910	0.8400545	0.9393692	
4:S 1:observable	0.9235234	0.0304887	0.8382301	0.9656867	
5:Psi 1 to U	0.1948037	0.0131660	0.1702848	0.2219089	
6:Psi 1 to U	0.3158615	0.0181433	0.2814219	0.3524485	
7:Psi 1 to U	0.2942561	0.0220380	0.2529769	0.3392124	
8:Psi U to U	0.5000000	0.0000000	0.5000000	0.5000000	Fixed
9:Psi U to U	0.1981535	0.0375785	0.1345362	0.2820494	
10:Psi U to U	0.3702709	0.0533046	0.2730652	0.4792660	

It is also worth having a look at the estimated abundances $\hat{N}$:

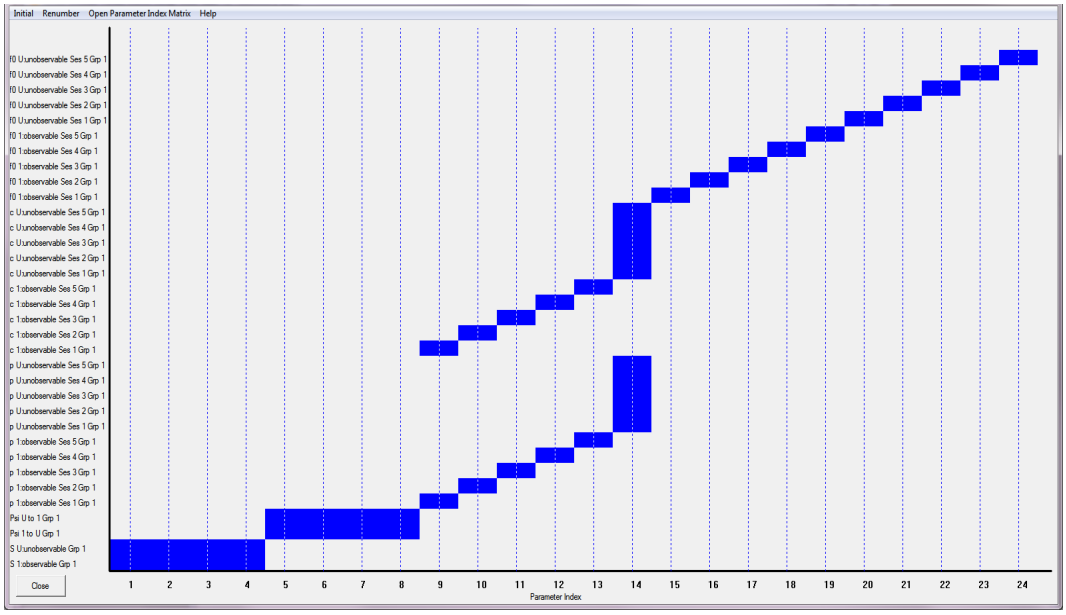


Since $M_{t+1} = 0$ for the unobservable state, then clearly $\hat{N}_{i,U} = 0$, as seen above.

For the '`Random movement`' model we do not need to constrain the last movement probabilities – movement is independent of which state you were in last time. So, we simply set $\psi^{OU} = \psi^{UU}$ (i.e., $\psi_1^{OU} = \psi_1^{UU}, \psi_2^{OU} = \psi_2^{UU}, \psi_3^{OU} = \psi_3^{UU}$...).

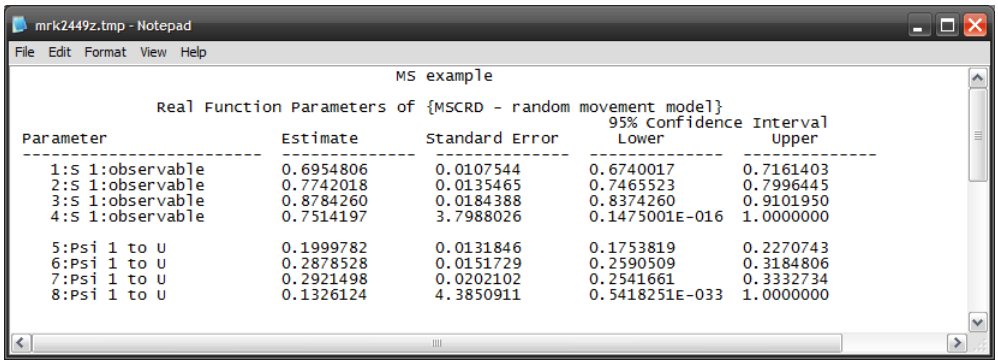


Here is the full PIM chart for the '`Random movement`' model.

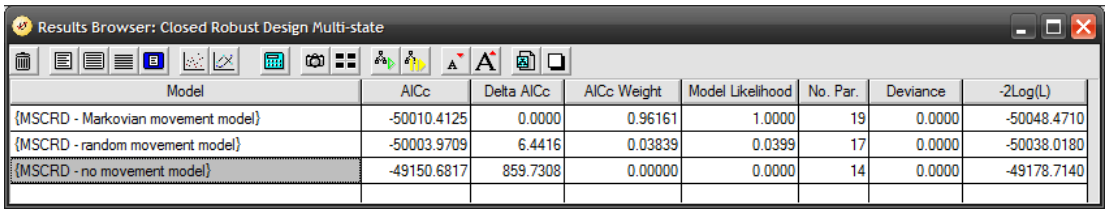


Go ahead and run this model – again fixing the encounter probability for the unobservable state to 0 first (note that for this model, this corresponds to parameter 14). Finally, recall that as was the case for the '`Markovian movement`' model, ψ_1^{UU} (parameter 5) does not enter the likelihood, but we constrain it for consistency. However, here we do not want to fix it to any value, since that would also imply fixing $\psi_1^{1u} = 0$, which we don't necessarily want to do.

Here are the results from fitting the MS '`Random movement`' model to the data – if you compare these estimates to those from the classic RD using 'γ' notation (p. 20), you'll see they are again virtually identical.



Finally, for the '`No movement`' model we fix $\psi_t^{OU} = \psi_t^{UU} = 0$, using the same PIM setup as the '`Random movement`' model (above). Here is the results browser with all three models:



Model	AICc	Delta AICc	AICc Weight	Model Likelihood	No. Par.	Deviance	-2Log(L)
{MSCRD - Markovian movement model}	-50010.4125	0.0000	0.96161	1.0000	19	0.0000	-50048.4710
{MSCRD - random movement model}	-50003.9709	6.4416	0.03839	0.0399	17	0.0000	-50038.0180
{MSCRD - no movement model}	-49150.6817	859.7308	0.00000	0.0000	14	0.0000	-49178.7140

Compare it against its counterpart from section 15.6.1 (p. 21), and you'll see they are identical. This drives home the point that the '**`Robust Design`**' and '**`MSCRD`**' options of **MARK** invoke two models that produce basically the same estimates. For the special case we have set up they represent the *exact* same likelihood, which reinforces the point that the '**`Robust Design`**' option represents a special case of the more general '**`Closed Robust Design Multi-state`**'.

In conclusion, if you can keep straight the definitions of the γ's and their relationship with ψ's, then for the case of one observable and one unobservable state, you can see from the examples we've shown that the '**`Robust Design`**' option is simpler to set up and deal with. The '**`Closed Robust Design Multi-state`**' option in **MARK** provides a more powerful and flexible tool for more complex scenarios that arise.

begin sidebar

the 'even flow' model

Back in section 15.3.4, we introduced a model we referred to as an '`even flow`' model. In the '`even flow`' model, we are interested in whether the probability of moving from 'observable' at time i to 'unobservable' at time $i+1$ is the same as the probability of moving from 'unobservable' to 'observable'

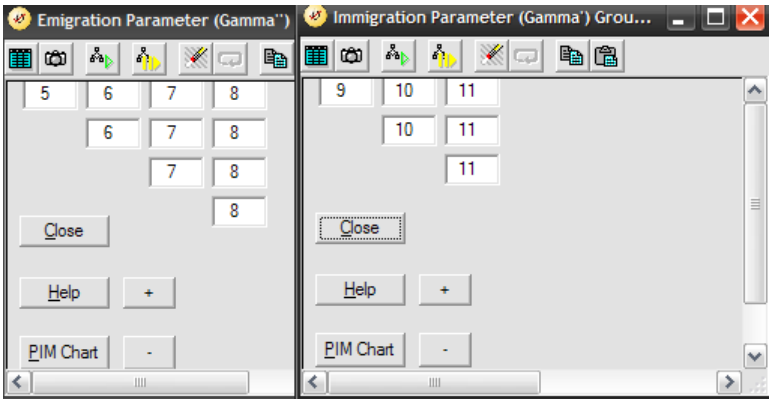
over the same interval. In other words, the the '`even flow`' model is specified by setting (i) $(1-\gamma') = \gamma''$ in the classic closed RD, which is equivalent to (ii) setting $\psi^{UO} = \psi^{OU}$ in the multistate closed RD (MSCRD).

Let's consider the MSCRD formulation first. We'll used the simulated data set we analyzed before (`ms_rdsimple1.inp`; 5 primary periods). Recall from our earlier analysis of these data that we expect time-dependence in movement probabilities between observable and unobservable states. Thus, to construct the 'even flow' model, we set $\psi_1^{UO} = \psi_1^{OU}, \psi_2^{UO} = \psi_2^{OU}, \ldots, \psi_4^{UO} = \psi_4^{OU}$. We should recognize at this point that we set $\psi_1^{UO} = \psi_1^{OU}$ for consistency only, since ψ_1^{UO} is undefined, and ψ_1^{OU} isn't in the likelihood. Thus, we would typically ignore the estimates over the first interval.

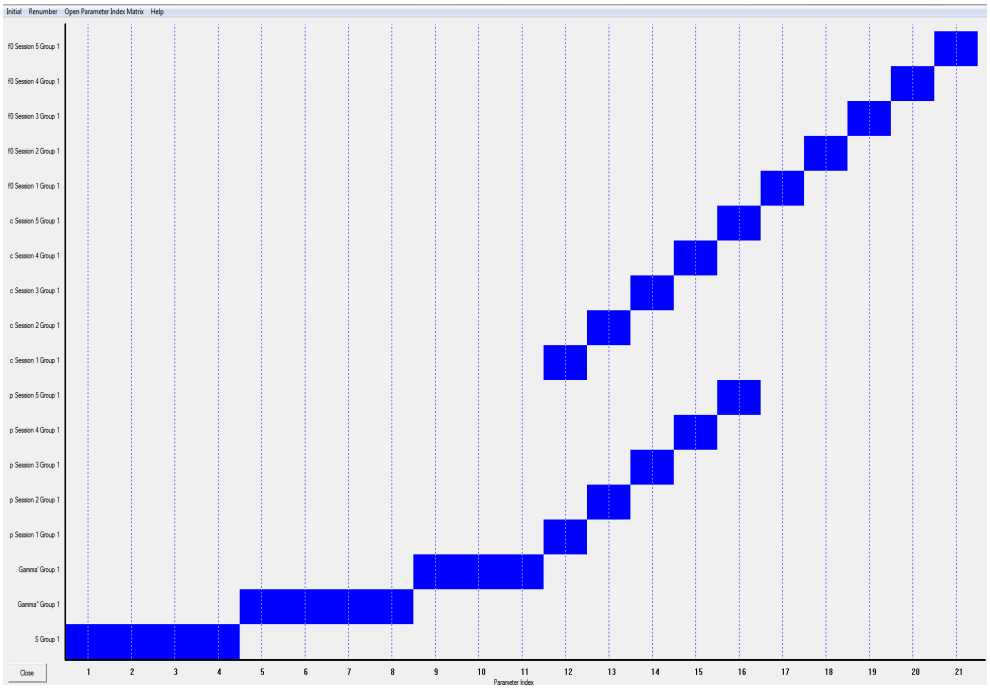
Bring up the results of your earlier MS analysis of these simulated data – retrieve the '`random movement`' model. Next, we'll want to change the PIM definitions, to make sure we have the parameters ψ^{UO} and ψ^{OU} in model. Select the '**`PIM | Change PIM definition`**' menu, and make sure you specify '`Psi 1 to 1`' and '`Psi U to U`' as the transition probabilities to obtain by subtraction. Once you've done so, stop and think for a moment. Do you need to do anything more to construct the 'even flow' model? No! you're done. In the 'random movement' model we retrieved, we'd set $\psi_i^{UU} = \psi_i^{OU}$. So, by simply changing the PIM definition so that ψ^{UU} is obtained by subtraction, then with the same PIM *structure* you retrieved from the '`random movement`' model, but now with different definitions for those parameters, you've already 'built' the '`even flow`' model. Go ahead an run this model, and add the results to the browser. You'll see that it doesn't do particularly well – better than the 'no movement' model, but much worse than the '`random movement`' or '`Markovian`' models.

So far – pretty straightforward. Perhaps unexpectedly so for the MSCRD approach, given the convenience of of switching directly from the '`random movement`' to '`even flow`' models simply by changing the PIM definition. However, what if instead we had used the classic 'γ parametrization of the closed RD? As noted earlier, fitting the '`even flow`' model using γ notation means setting $(1-\gamma') = \gamma''$. OK – fine. But how do you set this equality constraint, when the model is parameterized using only γ' and γ'', and where you are not able to specify that **MARK** used the complement of one of them? You can only 'change PIM definitions' for certain data types (such as the MS data type) – so how would would set $(1-\gamma') = \gamma''$? When we first introduced the '*Even flow*' model back in 15.3.4, we made some cryptic mention of a 'design matrix' trick that you would need to use in order to construct the model using the 'γ parameterizations. Time to introduce the 'trick'.

First, go back and re-open your classic closed RD analysis of `rd_simple.inp`. Retrieve the '`Markovian movement`' model in the browser. Open up the PIMs for γ' and γ'', and eliminate the Markovian constraints (in other words, make them both fully time-dependent, with no overlapping parameter indices). The PIM structure for the two parameters should now look like:



Here is the corresponding PIM chart for the 'random movement' model.



So, in effect, we've constructed a model with full time-dependence in S and both γ parameters.

For the next step, we want to build the DM corresponding to this 'fully time-dependent' model. At this point, this should be relatively straightforward for you. One version of a DM corresponding to this PIM chart is shown below:

B1:	B2:	B3:	B4:	Parm	B5: int - gamma	B6: t1	B7: t2	B8:	B9:	B10:	B11:	B12:	B13:	B14:	B15:	B16:	B17:	B18:	B19:
1	0	0	0	1:S	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	2:S	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	0	3:S	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	4:S	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	5:Gamma"	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	6:Gamma"	1	1	0	1	0	1	0	0	0	0	0	0	0	0	0
0	0	0	0	7:Gamma"	1	1	0	0	1	0	1	0	0	0	0	0	0	0	0
0	0	0	0	8:Gamma"	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	9:Gamma'	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	10:Gamma'	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0
0	0	0	0	11:Gamma'	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	12:p Session 1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	13:p Session 2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	14:p Session 3	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
0	0	0	0	15:p Session 4	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
0	0	0	0	16:p Session 5	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
0	0	0	0	17:f0 Session 1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
0	0	0	0	18:f0 Session 2	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	19:f0 Session 3	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
0	0	0	0	20:f0 Session 4	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
0	0	0	0	21:f0 Session 5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

(Note: we've used an identity matrix structure for the S, p and N parameters, since we are not particularly interested in them here.)

We do want to pay attention to the DM modeling of the γ parameter, obviously. Here, we've simply adopted the familiar 'intercept reference coding' approach we've used much of the time so far in this book. Recall that what we're doing here is specifying β terms as relative deviances from a common 'reference' value (i.e., the intercept).

Alternatively, we could use an *identity* coding scheme for the γ parameters, which would look like the following:

5:Gamma''	1	1	0	0	0	0	0	0
6:Gamma''	1	0	1	0	0	1	0	0
7:Gamma''	1	0	0	1	0	0	1	0
8:Gamma''	1	0	0	0	1	0	0	0
9:Gamma'	0	0	1	0	0	0	0	1
10:Gamma'	0	0	0	1	0	0	0	0
11:Gamma'	0	0	0	0	1	0	0	0

Both would yield identical real estimates on the normal probability scale – the difference between the two is in terms of interpretation of the β parameters in the linear model. In either case, note the similarity of the structure of the part of the DM coding the γ parameters to a typical 'age' model (Chapter 7) – reflecting the fact that there is no γ' parameter for the first interval (such that the coding for time for γ' starts with interval 2).

Now – the 'trick'. Not so much a 'trick', but rather a more advanced application of some DM concepts. We'll introduce the idea by first modifying our current time-dependent DM to specify the '`Random movement`' model (where $\gamma'_i = \gamma''_i$). Recall from earlier in this chapter that to build the '`Random movement`' model we applied a constraint to a DM with time-variation in the γ parameters (actually, we initially introduced it wrt to the '`Markovian model`', but the '`Markovian model`' is essentially time-dependent with some constraints on the terminal pair of γ parameters). What we do here depends on which form of the DM we're using for γ. We'll proceed as if we're using the identity DM for γ – it's somewhat easier to explain (as you'll see).

So, for a '`Random movement`' model, the DM corresponding to the γ parameters would look like

5:Gamma''	1	0	0	0
6:Gamma''	0	1	0	0
7:Gamma''	0	0	1	0
8:Gamma''	0	0	0	1
9:Gamma'	0	1	0	0
10:Gamma'	0	0	1	0
11:Gamma'	0	0	0	1

The modified DM if we'd used the intercept coding scheme would look like

5:Gamma''	1	1	0	0
6:Gamma''	1	0	1	0
7:Gamma''	1	0	0	1
8:Gamma''	1	0	0	0
9:Gamma'	1	0	1	0
10:Gamma'	1	0	0	1
11:Gamma'	1	0	0	0

Now – the 'trick. Here we've structured the DM for the '`Random movement`' model, which specifies that $\gamma'_i = \gamma''_i$. For the '`Even flow`' model, we want to specify that $(1 - \gamma'_i) = \gamma''$. In other words, we want so set the complement of γ'_i equal to γ''_i. Our current DM for the '`Random movement`' model is clearly pretty close, but how do we 'tell it' to use the complement of γ'_i?

In fact, it turns out that you can specify the equality of one parameter with the *complement* of another using a '`1, -1`' coding scheme, where we use the '1' to indicate one parameter, and '-1' to indicate the complement of the other. In this case, we would use '1' to code for $(1 - \gamma'$, and '-1' to code for γ''. Using the identity DM, we simply need to change the dummy coding for each time step using the '`1,-1`' coding convention (as described).

Here is the completed DM for the γ parameters, corresponding to the '`Even flow`' model:

5:Gamma''	1	0	0	0
6:Gamma''	0	-1	0	0
7:Gamma''	0	0	-1	0
8:Gamma''	0	0	0	-1
9:Gamma'	0	1	0	0
10:Gamma'	0	0	1	0
11:Gamma'	0	0	0	1

If you run this model, you'll see that if gives you exactly the same value for $-2\log(\mathcal{L})$ as we obtained for the '`even flow`' model using the MSCRD approach (above). What if instead of $(1-\gamma_i')=\gamma''$ we wanted $(1-\gamma_i'')=\gamma'$ (which would correspond to setting $\psi_i^{UU}=\psi_i^{UO}$)? Easy – simply reverse the DM coding so that '-1' is used for γ', and '1' is used for γ''.

Now, as a real test of your understanding, how would we modify the intercept-based DM for a '`Even flow`' model? The trick is to think – hard – about what the intercept represents. The 'answer' is shown below – no peeking! See if you can figure it out on your own. It's somewhat trickier than the identity matrix approach we demonstrated first, but if you understand what this modified DM shows (i.e., if you understand the 'strange' things we did to the intercept') then congratulations are in order, since that would exhibit a pretty solid understanding of the DM, and intercept coding in general.

5:Gamma''	-1	-1	0	0
6:Gamma''	-1	0	-1	0
7:Gamma''	-1	0	0	-1
8:Gamma''	-1	0	0	0
9:Gamma'	1	0	1	0
10:Gamma'	1	0	0	1
11:Gamma'	1	0	0	0

Nifty stuff, eh? Bonus points if you can figure out why this 'trick' works (i.e., what you're really doing with '1' and '-1' coding). Actually, if you understand the linear model being constructed, it isn't too bad.

end sidebar

15.8. The 'open' robust design...

Our discussion here of the robust design assumes that the closure assumption within a primary period is valid. In Chapter 14 we outlined conditions, discussed in Kendall (1999), where certain types of violation of closure do not induce bias in estimators. These same conditions are directly applicable here as well. However, one situation where this will not work well is where both arrivals and departures from the study area are occurring throughout the primary period. This situation falls under the umbrella of an '*open robust design*, which we describe here.

15.8.1. Background

The '**`Open Robust Design multi-state`**' data type in **MARK** (hereafter, MSORD) derives from Kendall & Bjorkland (2001, Biometrics) and Kendall & Nichols (2002, Ecology), based on the design first described by Schwarz & Stobo (1997, Biometrics). We'll use the case of nesting sea turtles from Kendall & Bjorkland (2001) to motivate the use of this data type, as well as how to use it. Schwarz & Stobo (1997)

used the case of a rookery of grey seals, and we also believe it could be quite useful for stopover studies of migratory species.

In a study of sea turtles there is an interest in estimating survival probabilities, breeding probabilities, and perhaps population size (as well as population growth rate). Nesting seasons are extensive, up to five months. Capture effort is typically throughout the season, in some cases nightly. Because sea turtles often lay more than one clutch, there is an opportunity to recapture a given female multiple times in a season. In summary, sampling for a given year consists of multiple sampling periods, where each individual in the nesting population has a chance (assumed to be the same chance) of being captured in each sampling interval. With a couple of additional assumptions, this constitutes a robust design.

In the preceding sections of this chapter, we described the closed robust design, where it was assumed that, for the duration of capture effort within a primary period, one of the following was true: (1) the population occupying the study area was completely closed to additions or deletions, (2) individuals moved completely randomly in and out of the study area, (3) all individuals were present in the first sampling occasion within a primary period, although marked and unmarked individuals could exit the study area (with the same probability) before the last sampling occasion for that season, or (4) individuals could enter the study area between the first and last sampling occasion within a season, assuming all individuals are present by the last sampling occasion. An additional assumption for conditions 3 and 4 is that capture probability within a primary period varies only by time (not trap response or individual heterogeneity).

In the case of nesting sea turtles, or marine mammal rookeries, the above assumptions do not hold. In fact, turtles arrive to lay their first clutch in a staggered fashion, remain in the area to renest for variable periods of time, then complete nesting and return to foraging areas in a staggered fashion. In essence, there is an open population study going on within each nesting season. First arrival at the nesting beach is equivalent to birth, and departure for the foraging grounds after the last clutch is laid is equivalent to death in a modeling sense.

If each individual in the population could be relied upon to be on the nesting beach each year, then the data for the entire nesting season could be pooled into whether or not an individual was captured in year t. However, some individuals skip nesting in a given year, and therefore the nesting population and population of female breeders in a given year are not equivalent. If nesting were a completely random process (i.e., each adult female had the same probability of nesting), then a CJS analysis from pooled data would produce an unbiased estimator for survival, although breeding probability could not be estimated. With most species, however, breeding probability is more accurately characterized as a Markov process (i.e., the probability of breeding is dependent on whether or not an individual is currently a breeder), and for some species skipping at least one year after breeding is obligatory. In this case, if skipping is not accounted for, all estimators in the CJS model, including survival, will be biased.

15.8.2. The General ORD Model

The essence of any robust design model is to take advantage of multiple sampling periods over a sufficiently short period of time that the state of the individual (e.g., nester or non-nester) remains static, in order to estimate the effective capture probability for those that are observable in that primary period (e.g., nesters). Because of the possibility of some individuals occupying an unobservable state (i.e., away from the study area[s]), we use a multistate approach to model the capture process across primary periods. For modeling captures within a primary period we use a generalization of the Schwarz-Arnason (1996, Biometrics) version of the Jolly-Seber model (see Chapter 12). The details of this generalization will become apparent below, but basically the probability an animal remains in the study area from one sampling period to the next can be modeled as a function of time (as in the Schwarz-Arnason model)

or the number of sampling periods since it first arrived that season (i.e., its 'age' within the season).

The ORD model first conditions on the total number of individuals captured in primary period t. Given that an individual is captured at least once within primary period t, the model then considers the probability of each observed capture history within that primary period. For example, if a nesting turtle is first captured on capture occasion 2 (of 6) of year t, the model considers two possibilities. She could have arrived to lay her first clutch during the first capture occasion (with probability $pent_{t_1}$), was not captured on that occasion (with probability $1 - p_{t_1}$), then returned to nest again (with probability $\varphi_{t_{1,0}}$) where time subscript 1 indicates sampling period 1 and age subscript 0 implies this is the first clutch laid this season by this female) and was captured (with probability p_{t_2}) during capture occasion 2. Alternatively, she could have arrived to lay her first clutch during capture occasion 2 (with probability $pent_{t_2}$). So for six capture occasions within a year (i.e, one primary period), we have the following probability structure for the history `010111`:

$$[pent_{t1}(1-p_{t1})\varphi_{t10}\varphi_{t21}\varphi_{t32}\varphi_{t43}\varphi_{t54} + pent_{t2}\varphi_{t20}\varphi_{t31}\varphi_{t42}\varphi_{t53}]p_{t2}(1-p_{t3})p_{t4}p_{t5}p_{t6}$$

which can be rewritten as

$$pent_{t1}(1-p_{t1})\varphi_{t10}p_{t2}\varphi_{t21}(1-p_{t3})\varphi_{t32}p_{t4}\varphi_{t43}p_{t5}\varphi_{t54}p_{t6}$$
$$+ pent_{t2}p_{t2}\varphi_{t20}(1-p_{t3})\varphi_{t31}p_{t4}\varphi_{t42}p_{t5}\varphi_{t53}p_{t6}$$

Because a turtle only arrives to lay her first clutch once, the entry probabilities ($pent_{ti}$) have to add to 1.0. Once captured within a year, subsequent captures within that year are modeled as a function of future 'survival' (in this case the probability a turtle keeps coming back to lay more clutches) and capture probability.

In summary , the following parameters will be found in the MSORD data type in **MARK**: S_t^r = survival from primary period t to $t+1$ for those occupying state r during period t; ψ_t^{rs} = probability an individual in state r in primary period t is in state s in primary period $t+1$, given it survives to period $t+1$; $pent_{tj}^s$ = probability that an individual in state s in primary period t is a new arrival (within that primary period) to the study area for that state at capture occasion j; φ_{tja}^s = probability that an individual in the study area associated with state s at capture occasion j, and who first arrived in the study area a capture occasions previous, is still in that study area at capture occasion $j+1$; and p_{tj}^s = probability that an individual in the study area for state s at capture occasion is captured. Of course any of these parameters can also be group- (e.g., sex) or true age-dependent.

Although useful and powerful, the use of the ORD in combination with MS models at least initially raises the dimensionality of the problem of programming models in **MARK**. As with the MSCRD model described in section 15.7, there are PIM's for state-specific survival between primary periods, and for state-specific transitions between primary periods. For each primary period there is a PIM for $pent$, φ, and p for each group and state, whereas for the MSCRD there were PIM's for p and c. The ORD also raises the dimensionality of model selection, where you explore variation in parameters both across primary periods (S, ψ, $pent$, φ, and p) and within primary periods ($pent$, φ, and p).

15.8.3. Implementing the ORD in MARK: simple example

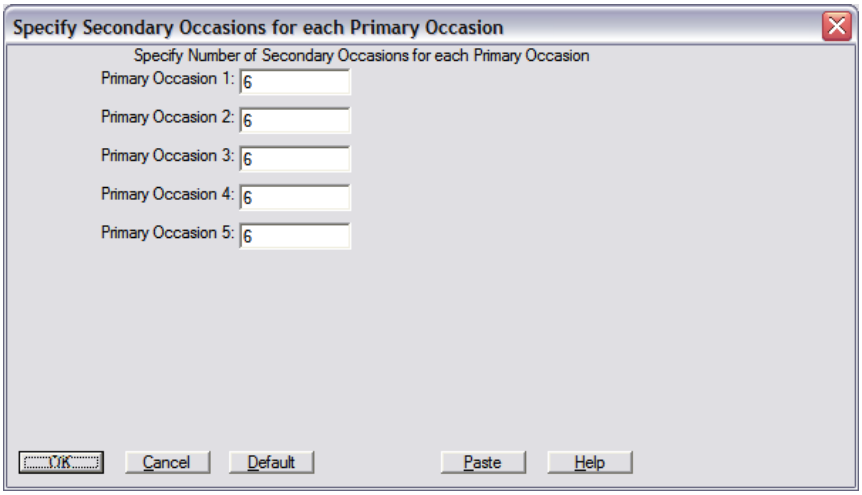
To illustrate some the points made above we will continue the example of nesting sea turtles on a single beach. We will consider a five-year study. Data are collected nightly on the beach in question, for three months. When laying multiple clutches, females space those clutches approximately two weeks apart. In dividing the season into capture occasions, it makes sense to do it so that each time an individual

re-nests she has a chance of being included in a capture occasion. Therefore we divide the three-month season into six half-month capture occasions (if an individual is captured one or more times in within the half-month interval you record a '1' in the capture history).

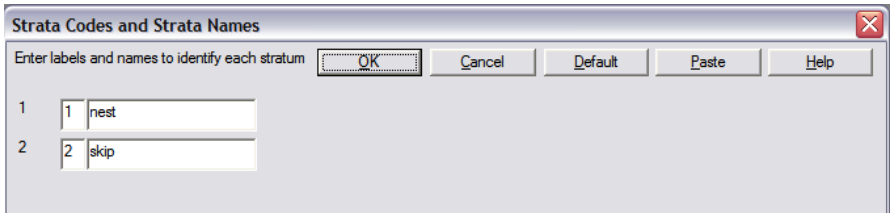
An example history for a five-year study, each with six capture occasions (totaling (5×6) = 30 capture occasions for the study) is

```
001010000000001111000000001001  1;
```

In this case a female is captured in sampling periods 3 and 5 in year 1, sampling periods 3-6 in year 3, and sampling periods 3 and 6 in year 5. As with other models in **MARK**, you provide the total number of encounter occasions (in this case 30). As with the closed robust design, when you designate the MSORD option in **MARK**, it provides a screen titled '**Easy Robust Times**', which is an aid to specifying how the capture history should be broken up into primary periods and capture occasions. **MARK** will ask how many primary occasions there are (in this case 5). **MARK** will then provide a screen indicating equal numbers of capture occasions per primary period. However, the MSORD model does not require them to be equal, and **MARK** allows you to correct these values.

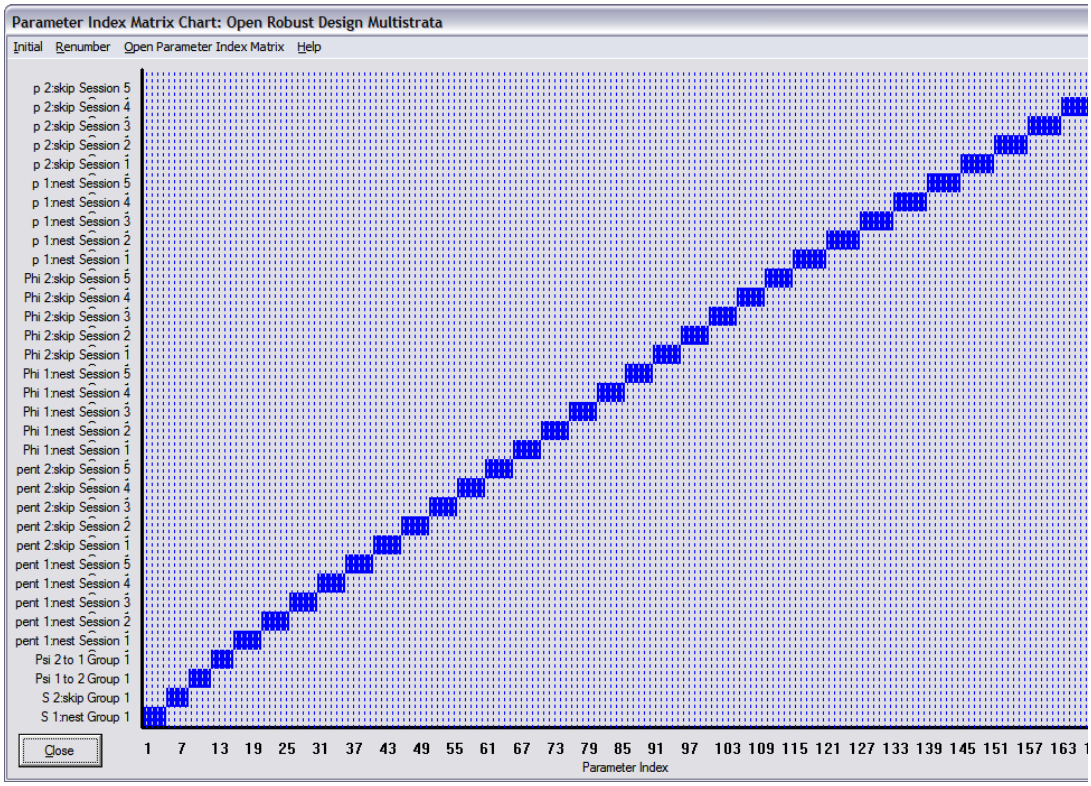


As with the '**Multi-state Recaptures only**' (Chapter 10) feature of **MARK**, after specifying the number of states you go to the '**Enter State Names**' screen, where you designate the label (the code used in the encounter history to designate the state), and name for each state. In the case of the example capture history above, where we have a single-site study, with one unobservable state, we would replace the **MARK** default of 'A' with a '1' in the label (to be consistent with the encounter history shown above), and might name it 'nest' (meaning that the animal was observed nesting). We can name the other state with something like 'skip' (because the animal was skipping nesting).



For this unobservable state it does not matter what label you give it (but do not make it '0' or the same as state 1), because animals are neither released nor observed in that state anyway.

When you have completed the model specification screen **MARK** will set up the PIM's for the '**MSORD**' model. Before we look at individuals PIMs, it's worth firing up the PIM chart, if for no other reason than to 'impress yourself' (and perhaps help convince your employers you need a bigger monitor). As noted earlier, PIM charts for robust design models can be 'busy' – the default time-dependent PIM chart for the turtle data is shown below.

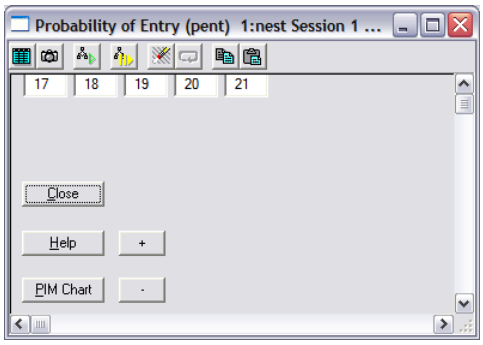


Pretty scary – it's so dense with information, you can barely read some of the labels on the left-hand side. As such, we'll do much of our manipulation of the basic parameter structure for our models using the individual PIMs.

Because this is a multi-state model, the PIM's for S and ψ are structured just as with the MS model (discussed at length in Chapter 10). They are upper triangular matrices, where you can specify these parameters to be constant, time-specific, partially age-specific, release-cohort specific, etc. You can also apply specific constraints to ensure transitions sum to 1. You can even specify which transitions are reported by **MARK**.

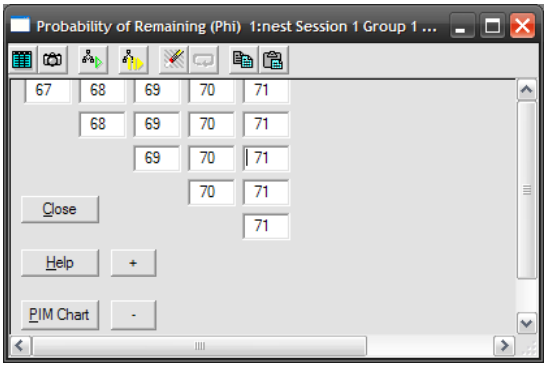
For parameters relating to capture histories within primary period, the PIM's for *pent* and p are really vectors, implying they can be modeled as a function of time or covariates, but not time since capture within the primary period.

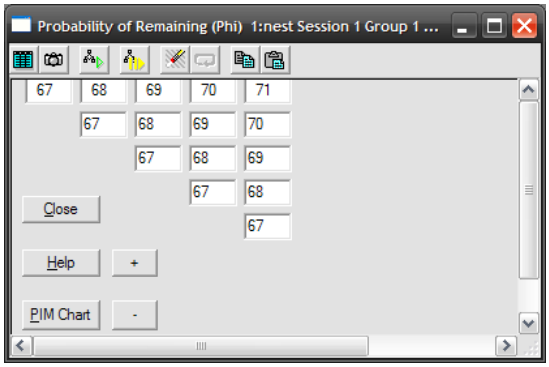
For example, here is the PIM for $pent_{t_1}$:



The PIM's for φ are of the typical format (i.e., an upper triangular matrix). However, keep in mind that typically the rows of a PIM denote a capture cohort, thereby permitting a parameter to be modeled as a function of time since first capture. For the φ PIM's the rows denote an *entry* cohort, permitting one to model these parameters as a function of the time since arrival to the study area (e.g., for nesting turtles that probability a female lays another clutch is a function of the number of clutches she has laid so far).

We provide two examples of PIM's below.



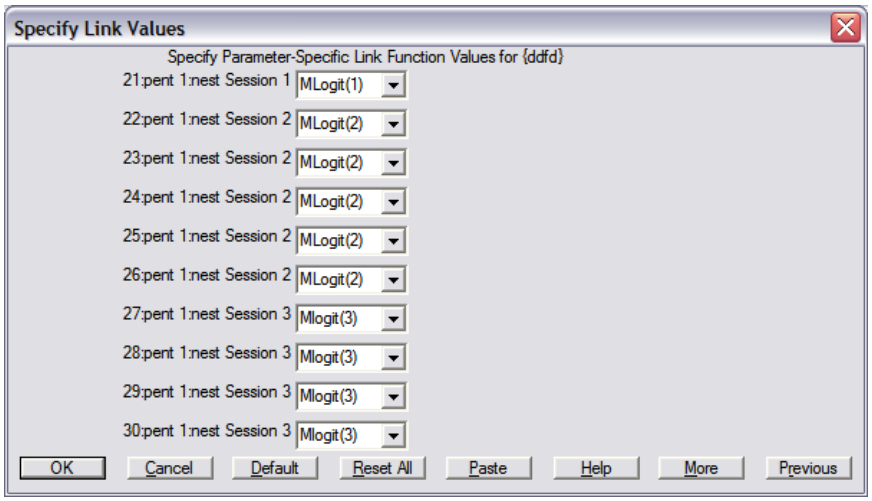


Whereas in the first (left-most) example φ is modeled strictly as a function of capture occasion, in the second example (right-most) it is modeled strictly as a function of the number of capture occasions since first arrival.

In the first PIM parameter 68 refers to the probability that an animal in the study area at capture occasion 2 will still be in the study area at capture occasion 3, independent of whether the animal was present or not present on occasion 1. In the second PIM above parameter 68 refers to the probability that an animal in the study area at any capture occasion j will still be in the study area at capture occasion $j+1$, given that the animal has been in the study area for two capture occasions, implying it first entered the study area that season at capture occasion $j-1$ (e.g., with sea turtles the individual laid her second clutch at capture occasion j).

There is an important point to consider about *pent*, the probability an animal arrives at the study area before any given sampling period, given that it arrives at some time during the season. For a given primary period the probability of entry across all sampling periods must add to 1.0 (i.e., $\sum_{j=1}^{l_t} pent_j = 1.0$). **MARK** derives $pent_{t1}$ by subtraction, and *therefore you cannot model this parameter directly*. In order

to satisfy this constraint reliably you should use the multinomial logit (`mlogit`) link in **MARK**, just as with the multistate models as described in Chapter 10. This is invoked in the '**Run**' screen by specifying the '**parm-specific**' option for the link function.

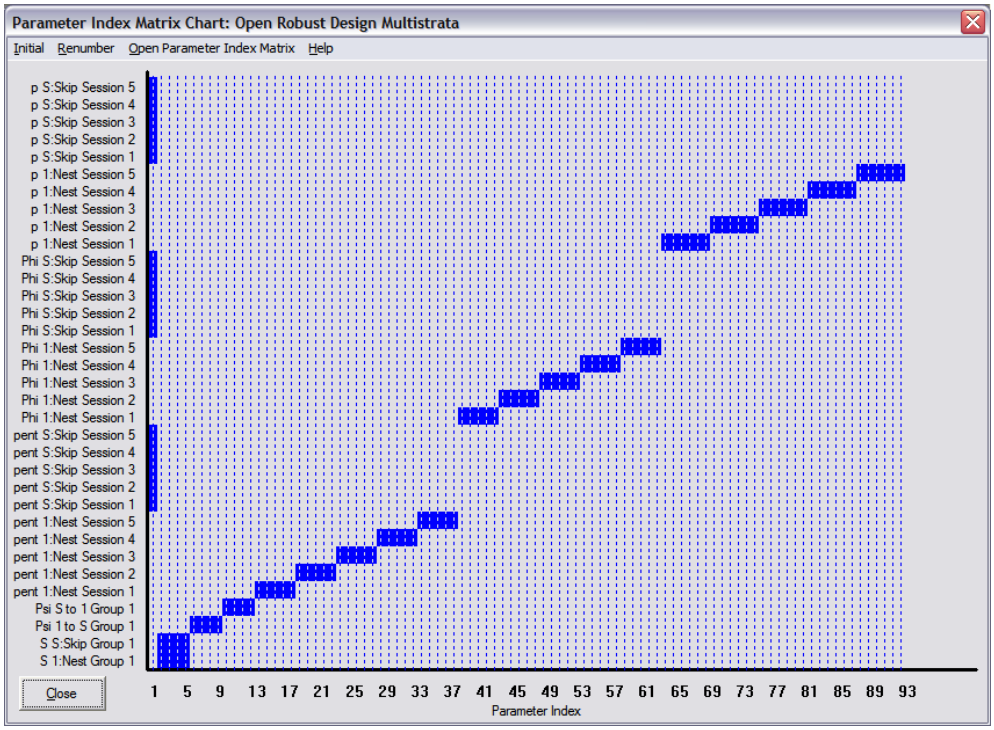


Each series of parameters that must add to one gets the same `mlogit` designation. For example, all *pent*'s for primary period 1 in state 1 would be assigned `mlogit(1)`, all *pent*'s for primary period 2 in state 1 would be assigned the `mlogit(2)` link, and so on. If you fail to do this you will most likely get an error message saying the numerical convergence was not reached. The entry probabilities are especially prone to this type of problem, because there are potentially so many different estimates that must sum to 1.0.

15.8.4. Dealing with unobservable states

Accounting for unobservable states with the '**MSORD**' feature of **MARK** is different than doing so in the original Robust Design option in **MARK** (discussed earlier in this chapter). With the latter, the model is set up explicitly for the case of one study area plus temporary emigrants. The fact that temporary emigrants actually occupy an unobservable state is treated implicitly. That model includes one PIM for survival (assumed the same for those in the study area and those outside), two PIM's for the temporary emigration process (coming and going), and one PIM for each primary period for detection probabilities in the study area. Conversely, the ORD model is nested within a general MS model framework. Therefore there will be PIM's for the within-primary-period parameters (*pent*, φ, and p) for each state. For a **T**-primary period study involving **V** states and **G** groups, this implies there will be $\mathbf{G}(\mathbf{V} \times 2 + \mathbf{V} \times \mathbf{T} \times 3)$ PIM's.

This framework is very flexible for dealing with unobservable states, because *an unobservable state is simply one where capture probability is always 0*. However, because of this flexibility there is also some irritation involved with dealing with all those PIM's, many of which do not get used. The PIM chart shown at the top of the next page illustrates a model from our example of one adult female population of sea turtles, where there is one observable (nesting) and one unobservable (skipping) state.



First, this chart illustrates yet again how this model can quickly make working directly with the PIM chart impractical (there are 34 PIM's in this relatively simple case of 5 years, 2 states, and 1 group). Second, it illustrates how within-primary-period parameters for the unobservable state are dealt with. Here we have set the *pent*'s, φ's, and p's for the unobservable state equal to 0, after assigning all these parameters to parameter index 1 (because these parameter will be set to 0 for each model considered, assigning them index 1 prevents you from being required to fix a different parameter to 0 for each run). Fixing the p's for the unobservable state to 0 is most important, because this implies the effective capture probability for the primary period will be 0. Once you do that, it does not matter what value you give to the *pent*'s or φ's because they never enter into the model as the animal is 'uncatchable' (i.e., unobservable in this state).

Also, note that we have set the survival PIM for the skipped breeders equal to that for the nesters. This is done implicitly in the original Robust Design model, but is necessary to do explicitly here as well. This constraint makes sense, since one cannot directly monitor survival of the unobservable animals, because they cannot be captured and released in the unobservable state. In general, it is a price to be paid for the fact that an unobservable state creates missing cells of data. However, under the assumption that survival is the same for both states, there is enough indirect information (from marked animals leaving and coming back) to estimate the transition probabilities ψ .

15.8.5. Which parameters can be estimated?

Identifying which parameters can be estimated can be a tricky business with these models, as it is for the other models in **MARK**. The first question is which parameters can be estimated based on the structure of the model (assuming no sparseness in the data). This issue is discussed for a single observable state in Kendall and Pollock (1982), Kendall *et al*. (1997), Kendall and Bjorkland (2001), Kendall and Nichols (2002), Fujiwara and Caswell (2002), and Schaub *et al*. (2004). If there are no unobservable

states, then under the ORD all S's and ψ's, for each time period and state, can be estimated (i.e., since the effective capture probability is estimable from the within-primary-period data, there is no confounding of parameters in the last period). For the case of one observable and one unobservable state, under a robust design, if $\psi_{T-1}^{(obs \rightarrow unobs)}$ and $\psi_{T-1}^{(unobs \rightarrow obs)}$ can be constrained to equal their counterparts for an earlier time period, then all the other parameters can be estimated as time specific. For multiple observable or unobservable states, investigations into estimability are ongoing, and call for methods such as described in Gimenez *et al*. (2004), and alternative numerical methods (see Appendix F) to investigate which parameters or combinations of parameters can be estimated, given the structure of the model. There are also some parameters within a primary period that cannot be estimated under the most general models. For the case where *pent*, φ, and p are all time dependent, $pent_{t1}$, p_{t1}, and $pent_{t2}$ are all confounded, as are φ_{tl-1}, p_{tl}, and $pent_{tl}$ (Kendall and Bjorkland 2001).

15.8.6. Goodness of Fit

As with other CMR models, there is no perfect answer to the question of how to assess absolute fit of the MSORD model to your data set. The only test specific to this model, for the case of one observable and one unobservable state, is a Pearson χ^2 test (pooling cells with small expected frequencies) available in program **ORDSURVIV** (Kendall and Bjorkland 2001, `www.pwrc.usgs.gov/software`).

15.8.7. Derived parameters from information within primary periods

In addition to the parameters listed above that are part of the '**MSORD**' model, **MARK** also reports two other derived parameters for each state: (i) the number of animals in that state in that primary period, $\hat{N}_t^{*S}$; and (ii) the residence time (or stopover time), $\hat{R}_t^s$, the average number of secondary sampling periods that an individual spent in the study area for that state s in primary period t. For the nesting sea turtle example these parameters would be the number of individual females that nested on that beach in year t, and the average number of nests laid per female in year t. These parameters are derived in the following way. First, effective capture probability for the primary period (i.e., the probability an animal is observed 1 or more times during the primary period, denoted as p^*) would be the sum of the probabilities an animal is first captured on each secondary sampling occasion.

For example, for a three-occasion primary period in state s:

$$\begin{aligned} p_t^{*s} =& pent_{t1}^s [p_{t1}^s + (1 - p_{t1}^s)\varphi_{t10}^s p_{t2}^s + (1 - p_{t1}^s)\varphi_{t10}^s (1 - p_{t2}^s)\varphi_{t21}^s p_{t3}^s] \\ &+ pent_{t2}^s [p_{t2}^s + (1 - p_{t2}^s)\varphi_{t20}^s p_{t3}^s] \\ &+ pent_{t3}^s p_{t3}^s \end{aligned}$$

Abundance is then estimated as $\hat{N}_t^s = n_t^{*s} / \hat{p}_t^{*s}$, where n_t^{*s} is the total number of individuals captured in state s during a primary period.

Expected residence time as defined here is of the following form for three secondary sampling periods:

$$\begin{aligned} E(R_t^s) = & 1 \times [pent_{t1}^s (1 - \varphi_{t10}^s) + \text{pent}_{t2}^s (1 - \varphi_{t20}^s) + pent_{t3}^s] \\ &+ 2 \times [pent_{t1}^s \varphi_{t10}^s (1 - \varphi_{t21}^s) + pent_{t2}^s \varphi_{t20}^s] \\ &+ 3 \times pent_{t1}^s \varphi_{t10}^s \varphi_{t21}^s \end{aligned}$$

The form of indicates that residence time is in units of time that match the time scale of secondary sampling periods. In the case of sea turtles, Kendall and Bjorkland (2001) partitioned the season into

sampling periods of 2 weeks. In addition, for the case where the probability of remaining in the study area is a function of the time since the animal first arrived, it assumes that sampling effort begins as soon as animals are available for capture (e.g., as soon as the first turtle arrives to nest). Otherwise, an animal captured in the first sampling occasion will be treated as if it just got there (i.e., it will be assigned to 'age' class 0) when in fact it might have been there for several periods (e.g., a sea turtle that might have already laid two previous clutches). Similarly, if the last sampling period does not coincide with the last departure by an animal marked in that primary period, bias will also be introduced.

15.8.8. Analyzing data for just one primary period

You might be interested in focusing on analysis of just one primary period. One reason might be to estimate the abundance or residence time parameters discussed above. Another use for this approach is in model selection. Robust design models add a layer of complexity to model selection, because possible variation in parameters goes on both at the primary period and secondary period levels. One approach to simplifying this process is to at least partially partition model selection with respect to *pent*, φ , and p from model selection with respect to S and ψ . Regardless, if you are interested in analyzing data for a given primary period you have two choices. If you are willing to assume that φ is *not* a function of an animal's 'age' within the primary period, then you are dealing with a Schwarz-Arnason Jolly-Seber model, and you can use the **POPAN** option in **MARK** (Chapter 12).

Otherwise you need to 'trick' the MSORD model. To do this, pretend that you have a two-primary period study. The data you are interested in analyzing will constitute the first primary period, and you will create a dummy second primary period consisting of at least two capture occasions.

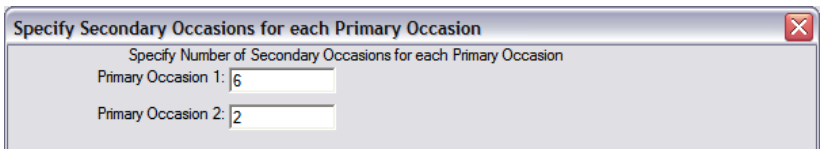
For example, if you want to analyze the following sea turtle capture histories for one primary period

```
111100
010010
111000
111100
011101
```

Then concatenate two more columns consisting of all 1's:

```
11110011  1;
01001011  1;
11100011  1;
11110011  1;
01110111  1;
```

Create an '**MSORD**' model in **MARK** as discussed before. In this case you specify 8 total encounters, and using the '**Easy Robust Times**' option you specify 2 primary periods, with 6 and 2 secondary samples, respectively.



You specify two states, as discussed above. Set up the PIM's as described above, with the following exception. In order for each of the animals captured in primary period 1 to have a history of '11' in primary period 2, each must have survived, returned to the study area in primary period 2, arrived in time for the first sample, stayed around for each of the two sampling occasions, and been captured each time. Therefore, for the observable state you would need to fix the S_1, $\psi_1^{obs \rightarrow obs}$, φ_{20}^{obs}, p_{21}^{obs}, p_{22}^{obs} and to 1.0 and $pent_{22}^{obs}$ to 0 (recall that is computed by subtraction). Maintain these constraints for each of the models you consider.

15.9. The robust design & unequal time intervals

As noted in Chapter 10, any data type with state transitions suffers from the same problem when the intervals between occasions are unequal (how **MARK** handles unequal intervals in general was introduced earlier in Chapter 4).

As introduced in Chapter 10, consider the case where an encounter occasion is missing in the multi-state data type. Consider the following valid **MARK** 5-occasion multi-state encounter history '`A.A00`', where the missing occasion is shown as a 'dot' and there are 2 states, `A` and `B`, and occasions are all 1 time unit apart. To explain this 'dot', several possibilities exist, namely:

$$S_1^A \psi_1^{AA} (1 - p_2^A) S_2^A \psi_1^{AA} p_3^A \ldots \quad \text{and} \quad S_1^A \psi_1^{AB} (1 - p_2^B) S_2^B \psi_2^{BA} p_3^A \ldots$$

However, suppose that you coded the data with the dot left out, and set the time intervals to 2, 1, and 1. That is, only 4 occasions are considered instead of 5. So the encounter history is now '`AA00`'. Unfortunately, this approach is going to give *very* different results from the proper parametrization above. **MARK** does not generate the probabilities for the transition to state `B` with this parametrization. The probability of surviving from occasion 1 to occasion 2 would now be $(S_1^S)^2$, with no consideration that the animal could have moved to state `B` during the missing occasion. So, even the survival estimates S will be incorrect. The ψ parameters for the first interval are not comparable to the ψ parameters for the second and third intervals because they represent different time scales.

Internally, within **MARK**, the time interval correction on S remains, but all time interval corrections from ψ have been removed. The motivating logic is that when time intervals are 'ragged', e.g., 1.1, 0.9, 1.05, 0.95, it may still make sense to apply a correction to S. However, this correction is inappropriate for ψ, and may even be questionable for S.

Given the deep connections between 'multi-state' models and 'robust design' models introduced in this chapter, it is perhaps not surprising that the same general issue applies here. Consider the robust design with 3 primary occasions, each with 2 secondary occasions. Assume that the data were not collected for the 2nd primary sample, giving an encounter history of '`11..11`'. The missing primary encounter history again can be explained by 2 possibilities:

$$\ldots S_1 \gamma_2'' S_2 (1 - \gamma_2') \ldots \quad \text{and} \quad S_1 (1 - \gamma_2'') (1 - p_2^*) S_2 (1 - \gamma_3'') \ldots$$

For the robust design data type, coding the encounter history as only 2 primary occasions, '`1111`', with time interval of 2 will give the correct parametrization for S (i.e., S^2), but as above, the γ' and γ'' parameters cannot be corrected with this simple trick because the possibility of leaving the encounter area is not considered. So, for robust design data types (including the multi-state robust designs), only survival rates are corrected with the time interval, but none of the transition parameters are corrected. Again, **user beware**! Think carefully about what unequal time intervals may be doing to your interpretation of the parameter estimates.

15.10. Literature

Cormack, R. M. 1964. Estimates of survival from the sighting of marked animals. *Biometrika*, **51**, 429-438.

Fujiwara, M., and H. Caswell. 2002. Temporary emigration in mark-recapture analysis. *Ecology*, **83**, 3266-3275.

Huggins, R. M. 1989. On the statistical analysis of capture-recapture experiments. *Biometrika*, **76**, 133-140.

Huggins, R. M. 1991. Some practical aspects of a conditional likelihood approach to capture experiments. *Biometrics*, **47**, 725-732.

Jolly, G. M. 1965. Explicit estimates from capture-recapture data with both death and immigration stochastic model. *Biometrika*, **52**, 225-247.

Kendall, W. L., and J. D. Nichols. 1995. On the use of secondary capture-recapture samples to estimate temporary emigration and breeding proportions. *Journal of Applied Statistics*, **22**, 751-762.

Kendall, W. L., K. H. Pollock, and C. Brownie. 1995. A likelihood-based approach to capture-recapture estimation of demographic parameters under the robust design. *Biometrics*, **51**, 293-308.

Kendall, W. L., J. D. Nichols, and J. E. Hines. 1997. Estimating temporary emigration using capture-recapture data with Pollock's robust design. *Ecology*, **78**, 563-578.

Kendall, W. L. and R. Bjorkland. 2001. Using open robust design models to estimate temporary emigration from capture-recapture data. *Biometrics*, **57**, 1113-1122.

Kendall, W. L., and J. D. Nichols. 2002. Estimating state-transition probabilities for unobservable states using capture-recapture/resighting data. *Ecology*, **83**, 3276-3284.

Kendall, W. L., and K. H. Pollock. 1992. The Robust Design in capture-recapture studies: a review and evaluation by Monte Carlo simulation. Pages 31-43 in *Wildlife 2001: Populations*, D. R. McCullough and R. H. Barrett (eds), Elsevier, London, UK.

Schaub, M., O. Gimenez, B. R. Schmidt, and R. Pradel. 2004. Estimating survival and temporary emigration in the multistate capture-recapture framework. *Ecology*, **85**, 2107-2113.

Schwarz, C. J., and W. T. Stobo. 1997. Estimating temporary migration using the robust design. *Biometrics*, **53**, 178-194.

Schwarz, C. J., and A. N. Arnason. 1996. A general methodology for the analysis of capture-recapture experiments in open populations. *Biometrics*, **52**, 860-873.

Seber, G. A. F. 1965. A note on the multiple recapture census. *Biometrika*, **52**, 249-259.

CHAPTER 16

Known-fate models

In previous chapters, we've spent a considerable amount of time modeling situations where the probability of encountering an individual is less than 1. However, there is at least one situation where we do not have to model detection probability – *known-fate data*, so-called because we *know* the fate of each marked animal with certainty. In other words, encounter probability is 1.0 (which must be true if we know the fate of a marked individual with certainty). This situation typically arises when individuals are radio-marked, although certain kinds of plant data can also be analyzed with the known fate data type. In such cases, known-fate data are important because they provide a theory for estimation of survival probability and other parameters (such as emigration). The focus of known fate models is the estimation of survival probability S, the probability of surviving an interval between sampling occasions. These are models where it can be assumed that the sampling probabilities are 1. That is, the status (dead or alive) of all tagged animals is known at each sampling occasion. For this reason, precision is typically quite high, as precise as the binomial distribution allows, even in cases where sample size is often fairly small. The only disadvantages might be the cost of radios and possible effects of the radio on the animal or its behavior. The model is a product of simple binomial likelihoods. Data on egg mortality in nests and studies of sessile organisms, such as mollusks, have also been modeled as known fate data.

In fact, the known fate data type is exactly the same as logistic regression in any statistical package. The main advantage of using **MARK** for known fate analysis is the convenience of model selection, and the capabilities to model average survival estimates easily, and compute random effects estimates.

16.1. The Kaplan-Meier Method

The traditional starting point for the analysis of known-fate data is the Kaplan-Meier (1958) – we'll discuss it briefly here, before introducing a more flexible approach that will serve as the basis for the rest of this chapter.

The Kaplan-Meier (hereafter, K-M) estimator is based on observed data at a series of occasions, where animals are marked and released only at occasion 1. The K-M estimator of the survival function is

$$\hat{S}_t = \prod_{i=1}^{t} \left(\frac{n_i - d_i}{n_i} \right)$$

where n_i is the number of animals alive and at risk of death at occasion i (given that their fate is known

at the end of the interval), d_i is the number known dead at occasion i, and the product is over i up to the tth occasion (this estimator is often referred to as the product-limit estimator). Critical here is that n_i is the number known alive at the start of occasion i and whose fate (either alive or dead) is known at the end of the interval. Thus, the term in parentheses is just the estimated survival for interval i. Note that n_i does not include individuals censored during the interval. It is rare that a survival study will observe the occasion of death of every individual in the study. Animals are 'lost' (i.e., censored) due to radio failure or other reasons. The treatment of such censored animals is often important, but often somewhat subjective. These K-M estimates produce a survival function (see White and Garrott 1990); the cumulative survival up to time t. This is a step function and is useful in comparing, for example, the survival functions for males vs. females.

If there are no animals that are censored, then the survival function (empirical survival function or ESF) is merely,

$$\hat{S}_t = \left(\frac{\text{number alive longer than } t}{n} \right) \quad \text{for } t \geq 0$$

This is the same as the intuitive estimator where no censoring is occurring: $\hat{S}_t = n_{t+1}/n_t$; for example, $\hat{S}_2 = n_3/n_2$. The K-M method is an estimate of this survival function in the presence of censoring. Expressions for the variance of these estimates can be found in White and Garrott (1990).

A simple example of this method can be illustrated using the data from Conroy *et al*. (1989) on 48 radio-tagged black ducks. The data are

	survived to occasion							
week	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
number alive at start	48	47	45	39	34	28	25	24
number dying	1	2	2	5	4	3	1	0
number alive at end	47	45	39	34	28	25	24	24
number censored	0	0	4	0	2	0	0	0

Here, the number alive at the start of an interval are to known be alive at the start of sampling occasion j. This is equivalent to being alive at the start of interval j. For example, 47 animals are known to be alive at the beginning of occasion 2. Forty-five are alive at the start of interval 3, but 4 are censored from these 45 because their fate is unknown at the end of the interval, so that $n_3 = 41$. A further example is that 34 ducks survived to the start of occasion 5. Thus, the MLEs are

$\hat{S}_1 = 47/48 = 0.979$
$\hat{S}_2 = 45/47 = 0.957$
$\hat{S}_3 = 39/41 = 0.951$ (*note*: only 41 because 4 were censored)
$\hat{S}_4 = 34/39 = 0.872$
$\hat{S}5 = 28/32 = 0.875$ (*note*: only 32 because 2 were censored)
$\hat{S}_6 = 25/28 = 0.893$
$\hat{S}_7 = 24/25 = 0.960$
$\hat{S}_8 = 24/24 = 1.00$

Here one estimates 8 parameters – call this model `S(t)`. One could seek a more parsimonious model in several ways. First, perhaps all the parameters were nearly constant; thus a model with a single

survival probability might suffice (i.e., `S(.)`) If something was known about the intervals (similar to the flood years for the European dipper data) one could model these with one parameter and denote the other periods with a second survival parameter.

Finally, one might consider fitting some smooth function across the occasions and, thus, have perhaps only one intercept and one slope parameter (instead of 8 parameters). Still other possibilities exist for both parsimonious modeling and probable heterogeneity of survival probability across animals. These extensions are not possible with the K-M method and K-L-based (i.e., AIC) model selection is not possible. To do this, we need an approach based on maximum likelihood estimation – as it turns out, the simple binomial model will do just that for known-fate data.

16.2. The Binomial Model

In the K-M approach, we estimated the survival probability by

$$\hat{S}_t = \prod_{i=1}^{t} \left(\frac{n_i - d_i}{n_i} \right)$$

where d_i is the number dying over the ith interval, and n_i is the number alive ('at risk') at the start of the interval and whose fate is also known at the end of the interval (i.e., not censored). Here, we use the equivalence (under some conditions) of the K-M estimator, and a binomial estimator, to recast the problem in a familiar likelihood framework.

Consider the situation for the case in which all animals are released at some initial time $t = 0$, and there is no censoring. If we expand the product term from the preceding equation, over the interval $[0, t]$,

$$\hat{S}_t = \left(\frac{n_0 - d_0}{n_0} \right) \left(\frac{n_1 - d_1}{n_1} \right) \cdots \left(\frac{n_t - d_t}{n_t} \right)$$

We notice that in the absence of censoring (which we assume for the moment), the number of animals at risk at the start of an interval is always the previous number at risk, minus the number that died the previous interval.

Thus, we can re-write the expanded expression as

$$\begin{aligned} \hat{S}_t &= \left(\frac{n_0 - d_0}{n_0} \right) \left(\frac{n_1 - d_1}{n_1} \right) \cdots \left(\frac{n_t - d_t}{n_t} \right) \\ &= \left(\frac{n_0 - d_0}{n_0} \right) \left(\frac{n_0 - (d_0 + d_1)}{n_0 - d_0} \right) \times \left(\frac{n_0 - (d_0 + d_1 + d_2)}{n_0 - (d_0 + d_1)} \right) \times \cdots \\ &\quad \times \left(\frac{n_0 - (d_0 + d_1 + \cdots + d_t)}{n_0 - (d_0 + d_1 + \cdots + d_{t-1})} \right) \end{aligned}$$

OK, while this looks impressive, its importance lies in the fact that it can be easily simplified to

$$\hat{S}_t = \left(\frac{n_0 - d_0}{n_0} \right) \left(\frac{n_0 - (d_0 + d_1)}{n_0 - d_0} \right) \times \cdots \times \left(\frac{n_0 - (d_0 + d_1 + \cdots + d_t)}{n_0 - (d_0 + d_1 + \cdots + d_{t-1})} \right)$$

$$= \left(\frac{n_0 - (d_0 + d_1 + \cdots + d_t)}{n_0} \right)$$

If you look at this expression closely, you'll see that the numerator is the number of individuals from the initial release cohort (n_0) that remain alive (i.e., which do not die – the number that die is given by $(d_0 + d_1 + \cdots + d_t)$), divided by the number initially released. In other words, the estimate of survival to time t is simply the number surviving to time t, divided by the number released at time 0.

Now, this should sound familiar – hopefully you recognize it as the usual binomial estimator for survival as (in this case) number of survivors ('successes', in a literal sense) in n_0 trials. Thus, if

$$\hat{S}_i = \frac{y_i}{n_i}$$

where y_i is the number surviving to time i (on the interval $[i-1, i]$), and n_i is the number alive ('at risk') at the start of the interval (i.e., at time i), then we can write

$$\hat{S}_t = \prod_{i=1}^{t} \left(\frac{n_i - d_i}{n_i} \right) = \prod_{i=1}^{t} \left(\frac{y_i}{n_i} \right)$$

If you recall the brief introduction to likelihood theory in Chapter 1 (especially the section discussing the binomial), it will be clear that the likelihood expression for this equation is

$$\mathcal{L}(\theta \mid n_i, y_i) = \prod_{i=1}^{t} S_i^{y_i} (1 - S_i)^{(n_i - y_i)}$$

where θ is the survival model for the t intervals, n_i is the number of individuals alive (at risk) during each interval, y_i is the number surviving each interval, and S_i is the MLE of survival during each interval.

As suggested at the start of this section, the binomial model allows standard likelihood-based estimation and is therefore similar to other models in program **MARK**. To understand analysis of known-fate data using the binomial model in **MARK**, we first must understand that there are 3 possible scenarios under the known fate model. In a known-fate design, each tagged animal either:

1. survives to end of study (detected at each sampling occasion so fate is known on every occasion)
2. dies sometime during the study (its carcass is found on the first occasion after its death so that its fate is known)
3. survives up to the point where its fate is last known, at which time it is censored → the fate is known

Note, for purposes of estimating survival probabilities, there is no difference between animals seen alive and then removed from the population at occasion k and those censored due to radio failure or for other reasons. The binomial model assumes that the capture histories are mutually exclusive and that animals are independent, and that all animals have the same underlying survival probability when individuals are modeled with the same survival parameter (homogeneity across individuals). Known fate data can be modeled by a product of binomials.

Let us reconsider the black duck data (seen previously), using the binomial model framework : $n_1 = 48$, and $n_2 = 44$; the likelihood is

$$\mathcal{L}(S_1 \mid n_1, n_2) = \binom{n_1}{n_2} S_1^{n_2} (1 - S_1)^{(n_1 - n_2)} = \binom{48}{44} S_1^{44} (1 - S_1)^{(48-44)}$$

Clearly, one could find the MLE, $\hat{S}_1$, for this expression (e.g., $\hat{S}_1 = 44/48 = 0.917$). We could also easily derive an estimate of the variance (see section 1.3.1 in Chapter 1). Of course, the other binomial terms are multiplicative, assuming independence. The survival during the second interval is based on $n_2 = 44$ and $n_3 = 41$,

$$\mathcal{L}(S_1 \mid n_1, n_2) = \binom{n_1}{n_2} S_2^{n_2} (1 - S_2)^{(n_1 - n_2)} = \binom{41}{44} S_2^{41} (1 - S_2)^{(44-41)}$$

As noted above, the likelihood function for the entire set of black duck data (modified to better make some technical points below) is the product of these individual likelihoods.

16.3. Encounter Histories

Parameterization of encounter histories for a known-fate data is critical, and is structurally analogous to the `LDLD` format used in some other analyses (e.g., Burnham's live encounter-dead recovery model) – these are discussed more fully in Chapter 2. For the encounter histories for known-fate data, each entry is paired, where the first position (`L`) is a 1 if the animal is known to be alive at the start of occasion *j*; that is, at the start of the interval. A '`0`' in this first position indicates the animal was not yet tagged or otherwise not known to be alive at the start of the interval *j* or else its fates is not known at the end of the interval (and thus the animal is censored and is not part of the estimation during the interval).

The second position (`D`) in the pair is '`0`' if the animal survived to the end of the interval. It is a '`1`' if it died sometime during the interval. As the fate of every animal is assumed known at every occasion, the sampling probabilities (p) and reporting probabilities (r) are 1. The following examples will help clarify the coding:

encounter history	*probability*	*interpretation*
`10 10 10 10`	$S_1 S_2 S_3 S_4$	tagged at occasion 1 and survived until the end of the study
`10 10 11 00`	$S_1 S_2 (1 - S_3)$	tagged at occasion 1, known alive during the second interval, and died during the third interval
`10 11 00 00`	$S_1 (1 - S_2)$	tagged at occasion 1 and died during the second interval
`11 00 00 00`	$(1 - S_1)$	tagged at occasion 1 and died during the first interval
`10 00 00 10`	$S_1 S_4$	tagged at occasion 1, *censored* for interval 2 and 3 (not detected, or removed for some reason), and re-inserted into the study at occasion 4
`00 00 10 11`	$S_3 (1 - S_4)$	tagged at occasion 3, died during the 4th interval
`10 00 00 00`	S_1	tagged at occasion 1, known alive at the end of the first interval, but not released at occasion 2 and therefore *censored* after the first interval

Estimation of survival probabilities is based on a release (1) at the start of an interval and survival to the end of the interval (0), mortality probabilities are based on a release (1) and death (1) during the interval; if the animal then was censored, it does not provide information about S_i or $1 - S_i$).

Some 'rules' for encounter history coding:

a. The two-digit pairs each pertain to an interval (the period of time between occasions).

b. There are only 3 possible entries for each interval:

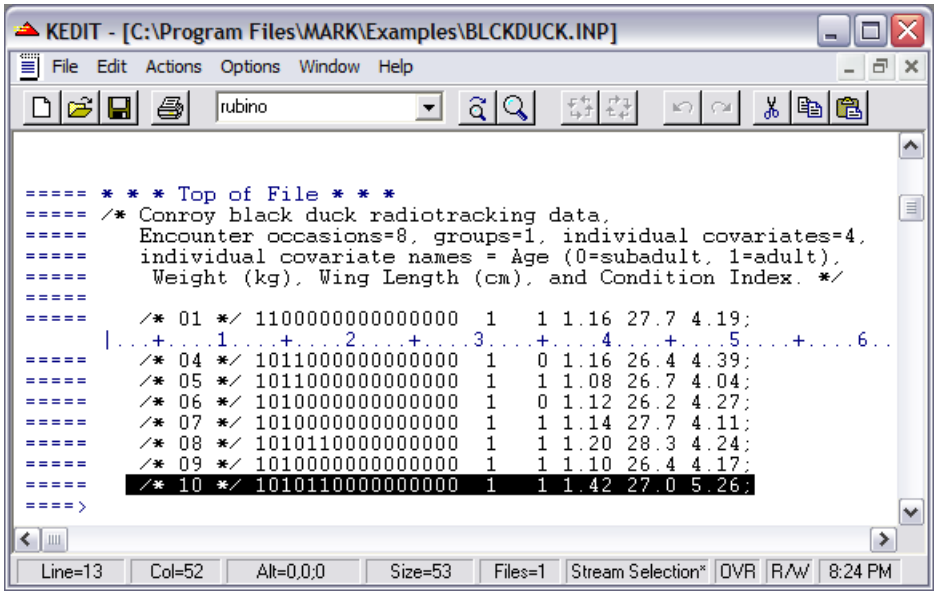
- `10` = an animal survived the interval, given it was alive at the start of the interval
- `11` = an animal died during the interval, given it was alive at the start of the interval
- `00` = an animal was censored for this interval

c. In order to know the fate of an animal during an interval, one must have encountered it **both** at the beginning **and** the end of the interval.

16.4. Worked example: black duck survival

Here, we consider the black duck radio-tracking data from Conroy *et al.* (1989). These data are contained in the `BLCKDUCK.INP` file contained in the **MARK** examples subdirectory that is created when you install **MARK** on your computer. The data consists of 50 individual encounter histories, 8 encounter occasions, 1 group, and 4 individual covariates: `age` (0 = subadult, 1 = adult), `weight` (kg), `wing` (length, in cm) and `condition`. In this study, it was suspected that variation in body size, condition (or both) might significantly influence survival, and that the relationship between survival and these covariates might differ between adult and subadult ducks.

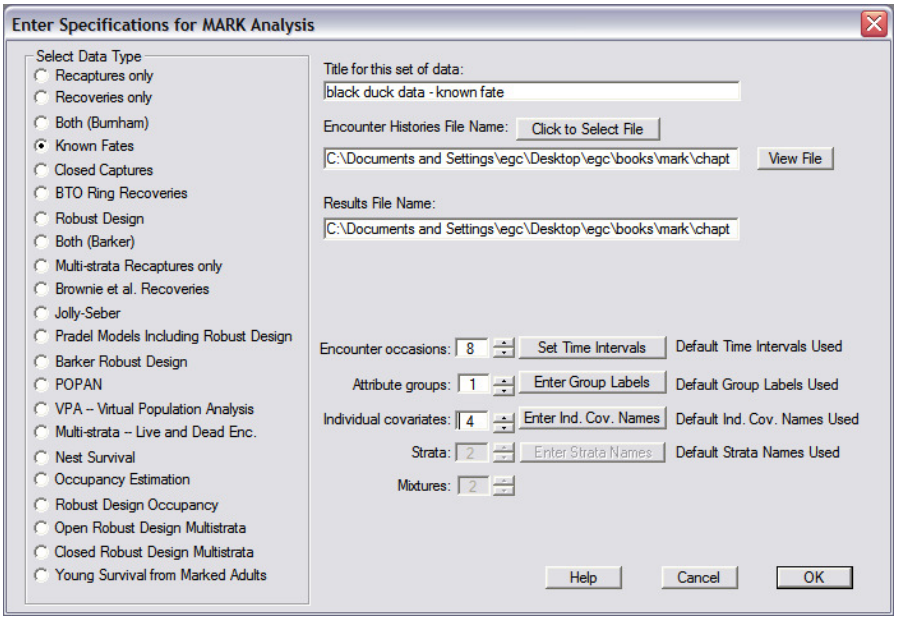
Here is what a portion of the `BLCKDUCK.INP` file looks like, showing the encounter histories and covariate values for the first 10 individuals:



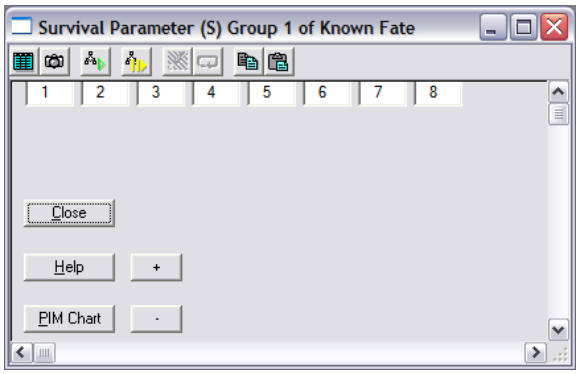
For example, the 10th individual in the data set has the encounter history '`1010110000000000`', meaning: marked and released alive at the start of the first interval, was detected alive at the start of the second interval, and then died during the third interval. The individual was radio-marked as an adult, and weighed 1.42 kilograms, had a wing length of 27.0 cm, and a condition index of 5.26. Ah – but look carefully – notice that in this `.INP` file, age is not coded as a classification variable (as is

typically done for 'groupings' of individuals – see Chapter 2), but instead as a dichotomous covariate. Coding dichotomous groups as simple linear covariates is perfectly acceptable – sometimes it is more straightforward to implement – the only 'cost' (for large data sets) might be efficiency (the numerical estimation can sometimes be slower using this approach). However, the advantage of coding age as an individual covariate is that if age turns out to be not important, then you are not required to manipulate PIMs for 2 age groups.

Obviously, this has some implications for how we specify this data set in **MARK**. Start a new project in **MARK**. Select '**known fates**' as the data type. Enter 8 encounter occasions. Now, the 'trick' is to remember that even though there are two age groups in the data, we're coding age using an individual covariate – as such, there is still only 1 attribute group, not two. So, leave attribute groups at the default of 1. For individual covariates, we need to 'tell' **MARK** that the input file has 4 covariates which we'll label as `age`, `weight`, `wing`, and `cond` (for condition), respectively.



Once we've specified the data type, we'll proceed to fit a series of models to the data. Let's consider models S_t, S_{age}, $S_.$, S_{weight}, S_{wing}, and S_{cond}. Clearly, the most parameterized of these models is model S_t, so we'll start with that. Here is the PIM for survival:

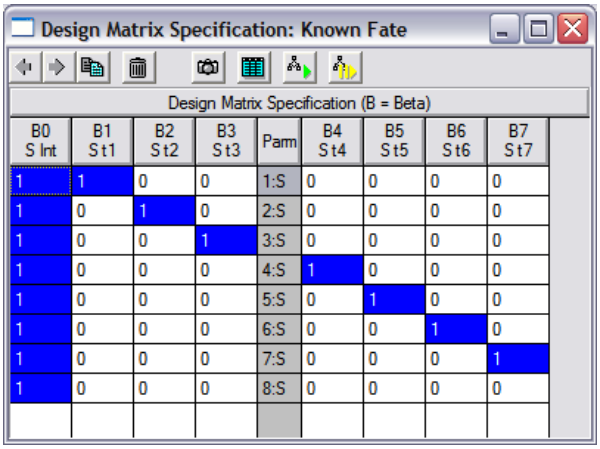


Not only is this particular PIM rather 'boring' (only a single row), in fact, there are no other PIMs for this analysis! Why? Easy – for known-fate data, we assume that all individuals are detected at each occasion, conditional on being alive and in the sample (i.e., we assume detection probability equals 1). Thus, the only parameter to model is the survival parameter (this should make sense – look back at the table on page 4 of this chapter – notice that the probability expressions corresponding to the different encounter histories are functions only of S_i – no encounter probability is included).

Why only a single row? Again, the assumption in the 'known-fate' analysis is that all individuals are released on the same occasion – presumably, at the start of the study (we'll consider staggered entry designs later). So, a single row, since all individuals in the analysis are in the same release cohort. Of course, this means that you aren't able to separate 'time' effects from 'age' effects in the analysis – at least, using the '**known fates**' data type in **MARK**. Remember, the age factor in this analysis is acting as a *classification* variable – and does not indicate the effects of aging (getting older over the course of the study) on survival. If you're marked individuals are all adults, then this may not be a particular problem. But, if your marked sample are subadults or young individuals, or perhaps a heterogeneous mixture of adults which might contain some proportion of transient individuals (see Chapter 7), you might have a problem. We'll deal with this later on in the chapter. For now, we'll proceed with the analysis, assuming all the assumptions of the classic known-fates analysis have been met.

Given the preceding discussion, it should be clear that for a known-fate data, the PIMs (and as a result, the model-fitting in general) is very straightforward. The default PIM (shown on the previous page) corresponds to model S_t. We go ahead, fit the model, and add the results to the browser. Recall that the default link function when using the PIM approach to model fitting is the sin link.

But, also recall that ultimately, we want to use the complete flexibility of the design matrix for fitting models in **MARK**. So, let's 're-build' our starting model S_t, using the design matrix. In this case, since the model we're fitting corresponds to the fully time-dependent model, we can generate the design matrix simply by selecting '**Design | Full**', which yields the following design matrix:



B0 S Int	B1 St1	B2 St2	B3 St3	Parm	B4 St4	B5 St5	B6 St6	B7 St7
1	1	0	0	1:S	0	0	0	0
1	0	1	0	2:S	0	0	0	0
1	0	0	1	3:S	0	0	0	0
1	0	0	0	4:S	1	0	0	0
1	0	0	0	5:S	0	1	0	0
1	0	0	0	6:S	0	0	1	0
1	0	0	0	7:S	0	0	0	1
1	0	0	0	8:S	0	0	0	0

Go ahead and fit this model, and add the results to the browser – label the model '`S(t) - DM`', to indicate it is the S_t model, constructed using a design matrix (DM) approach. Once you've added this model to the results browser (shown at the top of the next page), you'll notice that the two models (which are structurally identical) report different numbers of estimated parameters – 7 estimated parameters for the model fit using the DM (and the logit link), and 8 estimated parameters for the model fit using the PIM approach (and the sin link).

Results Browser: Known Fate

Model	AICc	Delta AICc	AICc Weight	Model Likelihood	No. Par.	Deviance
{S(t) - DM}	138.1008	0.0000	0.74248	1.0000	7	123.6950
{S(t) - PIM}	140.2186	2.1178	0.25752	0.3468	8	123.6950

In fact, what we see here is fairly common for known-fate studies – in many such studies, the sampling interval is often relatively short, such that the survival probabilities over each interval are often relatively close to 1.0. We discussed previously (Chapter 6) how the different link functions behave when parameter values are near the [0, 1] boundary. In the present example, examination of the reconstituted parameter values on the probability scale are in many cases close to the boundary – the two models differ in the estimate of survival for the last interval – the sin link estimates survival for the final interval at 1.00, whereas the logit link estimates the survival as 1.00, but fails to properly count this parameter as being estimated. We know that the number of estimated parameters for this analysis is 8 – so, we manually adjust the number of parameters for the model fit using the design matrix from 7 to 8 (when we do so, we see that the AIC_c and related statistics for the two models are now identical). We then delete the model fit using the PIM, since it is redundant to the model fit using the DM.

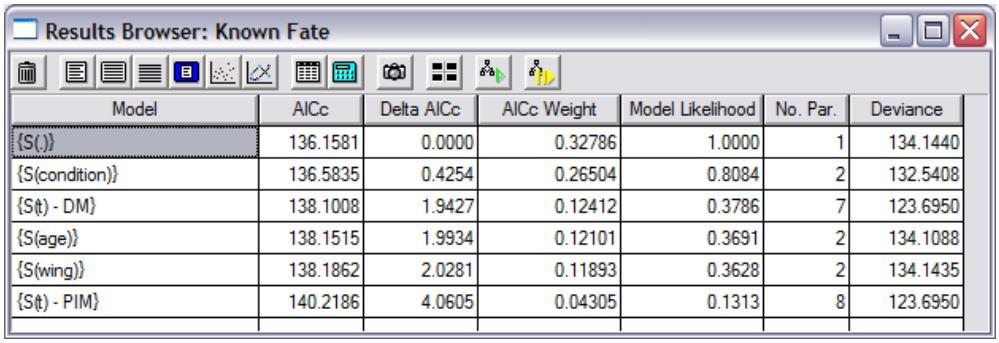
The next model in our candidate model set is model S_{age}. Recall that for this analysis, `age` is entered as a linear covariate in the `.INP` file, where `age` = 0 for subadults, and `age` = 1 for adults. Recall from Chapter 11 that individual covariates are introduced directly into the design matrix. So, for model S_{age}, the design matrix will look like

Design Matrix Specifi...

Design Matrix Specification (B = Beta)

B0	Parm	B1
1	1:S	age
1	2:S	age
1	3:S	age
1	4:S	age
1	5:S	age
1	6:S	age
1	7:S	age
1	8:S	age

With this design matrix, we can interpret β_2 as the difference in survival between subadults and adults, i.e., what should be added on a logit scale to the subadult survival estimate to obtain the adults survival estimate (interpretation of the β_i terms in the linear model is discussed at length in Chapter 6). We run this model, and add the results to the browser. We do much the same thing for each of the remaining models in the model set – each time, making simple modifications to the design matrix. The results browser showing the results from all of the models in our candidate model set is shown at the top of the next page. Interpretation and processing of the results follows the usual process outlined in earlier chapters, so we will not elaborate further here.



Results Browser: Known Fate

Model	AICc	Delta AICc	AICc Weight	Model Likelihood	No. Par.	Deviance
{S(.)}	136.1581	0.0000	0.32786	1.0000	1	134.1440
{S(condition)}	136.5835	0.4254	0.26504	0.8084	2	132.5408
{S(t) - DM}	138.1008	1.9427	0.12412	0.3786	7	123.6950
{S(age)}	138.1515	1.9934	0.12101	0.3691	2	134.1088
{S(wing)}	138.1862	2.0281	0.11893	0.3628	2	134.1435
{S(t) - PIM}	140.2186	4.0605	0.04305	0.1313	8	123.6950

16.5. Pollock's staggered entry design

The usual application of the Kaplan-Meier method assumes that all animals are released at occasion 1 and they are followed during the study until they die or are censored. Often new animals are released at each occasion (say, weekly); we say this entry is 'staggered' (Pollock *et al*. 1989). Assume, as before, that animals are fitted with radios and that these do not affect the animal's survival probability. This staggered entry fits easily into the K-M framework by merely redefining the n_i to include the number of new animals released at occasion i. Therefore, conceptually, the addition of new animals into the marked population causes no difficulties in data analysis.

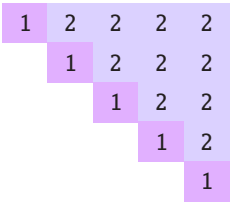
But, you might be wondering how you handled staggered entry designs in **MARK** – after all, how do you handle more than one cohort, if the survival PIM has only one row? If you think that the survival of the newly added animals is identical to the survival of animals previously in the sample, then you can just include the new animals in the encounter histories file with pairs of '`00`' `LD` codes prior to when the animal was captured and first put into the sample.

But what if you think that the newly added animals have different survival. Obviously, you need more rows. How? As it turns out, there is a straightforward and fairly intuitive way to tweak the known-fate data type (in this case, allowing it to handle staggered entry designs) – you simply add in additional groups for each release occasion (each release cohort), thus allowing cohort-specific survival probabilities. For this to work, you need to fix the survival probabilities for these later cohorts prior to their release to 1, because there is no data available to estimate these survival rates. With multiple groups representing different cohorts, analyses equivalent to the upper-triangular PIMs of the CJS and dead recovery data types can be developed.

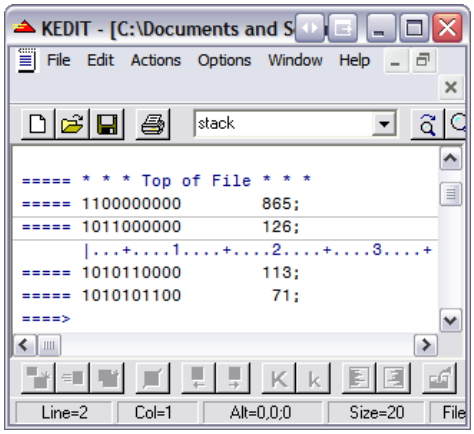
16.5.1. Staggered entry – worked example

To demonstrate the idea, we'll consider a somewhat complex example, involving individuals radio-marked as young – the complexity lies in how you handle the age-structure. Within a cohort, age and time are collinear, but with multiple release cohorts, it is possible to separate age and time effects (see Chapter 7). We simulated a dataset (`staggered.inp`) where individuals were radio-marked as young and followed for 5 sampling intervals – assume each interval is (say) a month long. We assume that all individuals alive and in the sample were detected, and that all fatalities were recorded (detected). We let survival in the interval following marking be 0.4, while survival in subsequent intervals (with a given cohort) was 0.8. For convenience, we'll refer to the two age classes as 'newborn' and 'mature', respectively.

If this were a typical 'age' analysis (see Chapter 7), this would correspond to the following PIM structure:

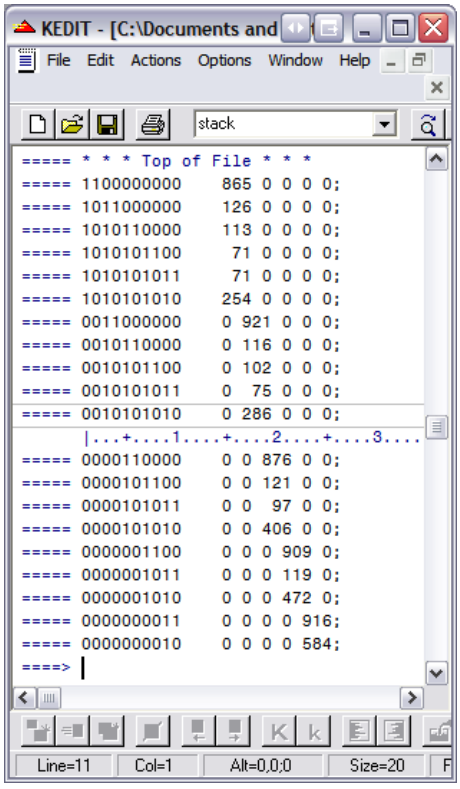


But, here we are considering a known-fate data, with staggered entry. To begin, let's first have a look at the `.INP` file – the first few lines are shown below:

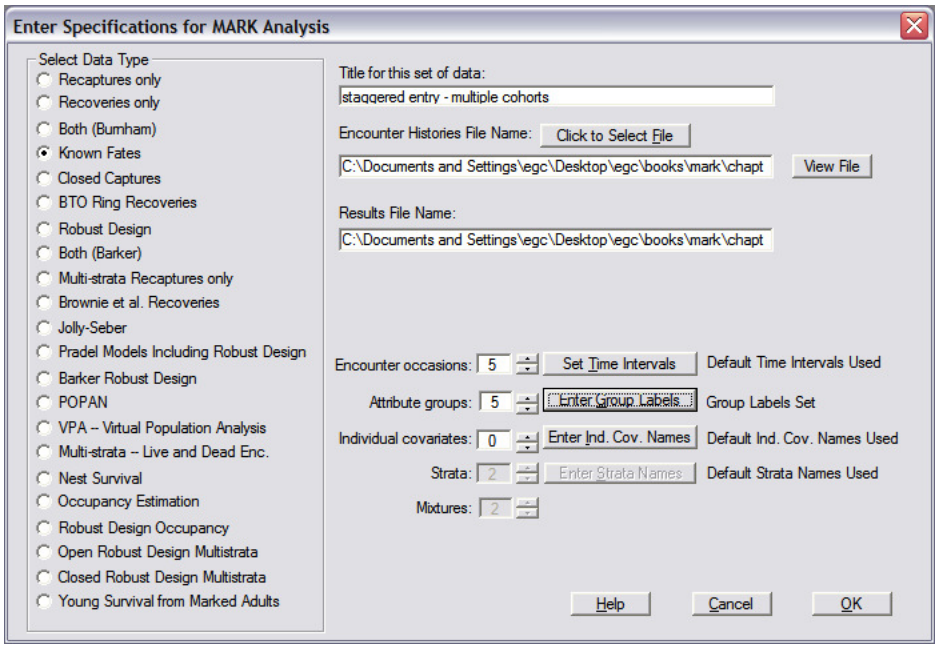


Now, at first glance, the structure of this file might seem perfectly reasonable. There are 5 occasions, in `LDLD` format. We see, for example, there were 865 individuals marked and released in the first cohort which died during the first interval (as newborns), 126 which were marked and released in the first cohort which died during the second interval (as mature individuals), and so on. But, what about the second cohort, and the third cohort, and so on?

How do we handle these 'additional cohorts'? As mentioned, we accomplish this in the known-fate data in **MARK** by specifying multiple groups – one group for each additional release cohort (in this case, 5 groups). However, while it is easy enough to specify 5 groups in the data specification window in **MARK**, we first need to modify the `.INP` file to indicate multiple groups. Recall from earlier chapters (especially Chapter 2), that each grouping requires a *frequency* column. So, 5 groups mean 5 frequency columns – not just the single frequency column we start with. The fully modified `staggered.inp` file is shown at the top of the next page (we'll let you make the modifications yourself). Notice that there are now 5 frequency columns – the first frequency column corresponds to number of individuals marked and released in cohort 1, the second frequency column corresponds to the number of individuals marked and released in cohort 2, and so on. Pay particular attention to the structure of these frequency columns.



Now that we've modified the `.INP` file (above), we can go ahead and run the analysis in **MARK**. We select the known-fate data type, and specify 5 groups (which we'll label as `C1`, `C2`,. . . ,`C5`, for cohort 1, cohort 2, and so on, respectively, to cohort 5):



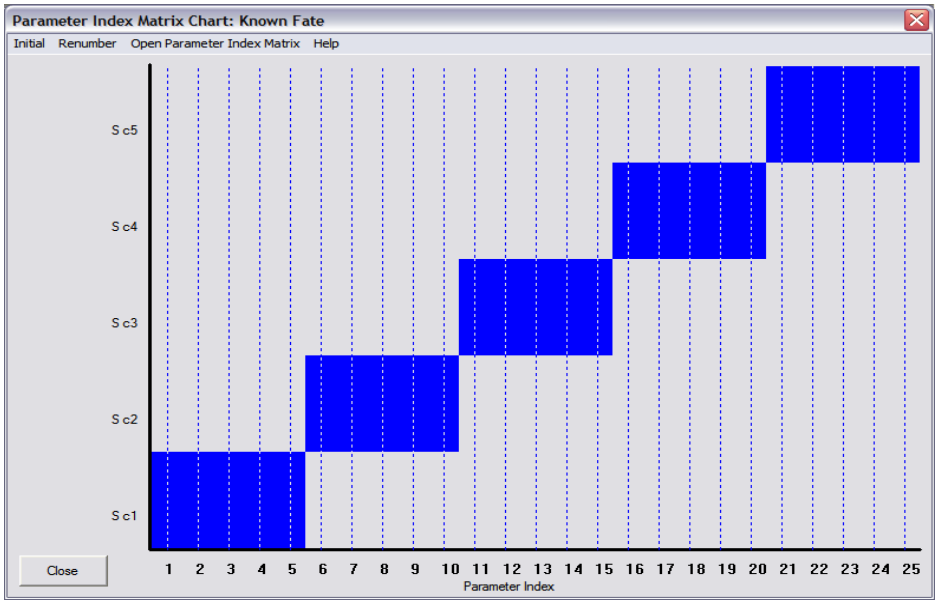
OK, so far, so good. Now for the only real complication – how to structure the PIMs for each cohort, and which parameters to fix to 1, in order for the analysis to make sense. Let's consider the following 2 models for our model set: $S_{a_2 \times cohort}$, and S_{a_2}. The first model indicates 2 age classes (newborn, and mature), with differences among cohorts. This corresponds to the following PIM structure:

1	6	6	6	6
	2	7	7	7
		3	8	8
			4	9
				5

The second model has differences in survival among the two age classes, but no differences among cohorts. This corresponds to the following PIM structure:

1	2	2	2	2
	1	2	2	2
		1	2	2
			1	2
				1

OK, so these are the 2 models we want to fit in **MARK**. The challenge is figuring out how to build them, and which parameters to fix. Clearly, the first model `S(a2 * cohort)` is the most general (since it has the most parameters), so we'll start there. Here is the default PIM chart for these data:



We see from the following PIM structure for this model that the first cohort consists of 2 age classes, as does the second, third, and so on. So, we might choose to simply right-click on the various 'blue-boxes' in the PIM chart, and select '**age**' – specifying 2 age-classes. Now, while you could, with some care, get this to work, there is an alternative approach which, while appearing to be more complex (and initially perhaps less intuitive), is in fact much easier.

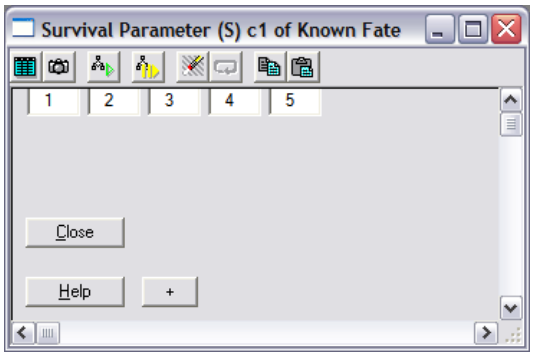
The key is in remembering that in the known-fates staggered entry analysis, we treat each cohort as if it were a separate group, fixing any '00' cells preceding the initial encounter in a cohort to 1.0. Again, keep in mind that each row (cohort) represents (analytically) a separate group. And, as noted, we want to fix the estimate for any of the preceding '00' cells to 1.0. Where do these cells occur? We've added them to the PIM in the following:

1	6	6	6	6
00	2	7	7	7
00	00	3	8	8
00	00	00	4	9
00	00	00	00	5

Now for the big step – if all of the '00' cells are ultimately to be fixed to 1.0, then we clearly would need only one parameter to code for them. So, let's rewrite the PIM, using the parameter 1 for the '00' cells, and then increasing the value of all of the other parameters by 1:

2	7	7	7	7
1	3	8	8	8
1	1	4	9	9
1	1	1	5	10
1	1	1	1	6

OK, now what? Well, each cohort is a group. So, we open up the PIM chart for each of the 5 groups (cohorts) in our example analysis – each of them has the same structure: a single line – here is the starting PIM for cohort 1:

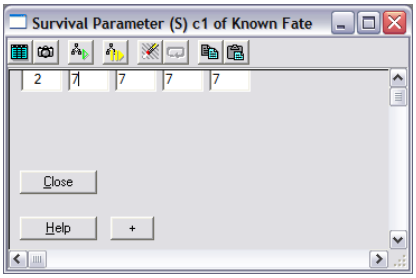


So, remembering that we want the overall PIM structure (over all cohorts) to look like

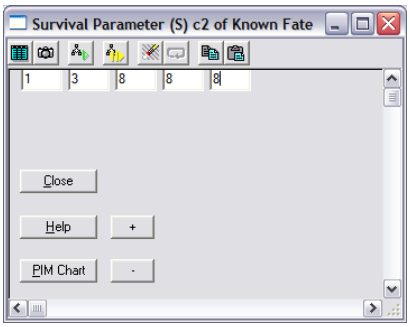
2	7	7	7	7
1	3	8	8	8
1	1	4	9	9
1	1	1	5	10
1	1	1	1	6

then it should be clear how to modify the PIM for cohort 1 - it needs to be modified to correspond to the first row of the overall PIM structure.

In other words, for cohort 1

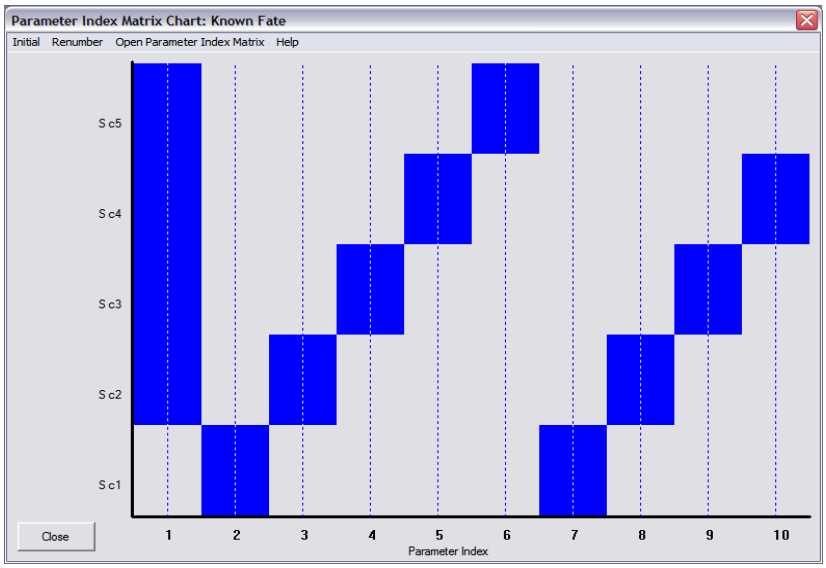


and for cohort 2,



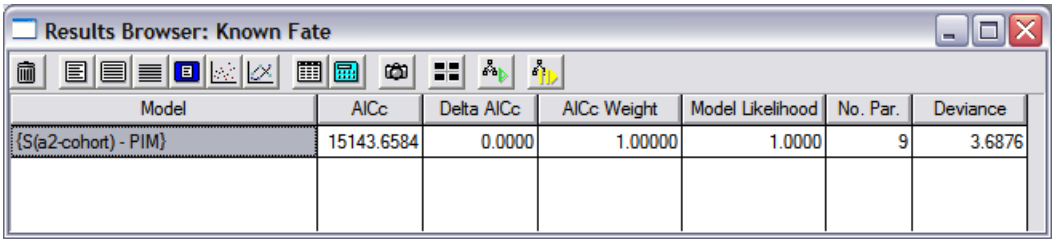
and so on – each PIM modified to match the corresponding row (representing a specific cohort) in the overall PIM.

Before we run our analysis, it's worth looking at the PIM chart for the model we've just created:



Note that the new parameter 1 occurs only in groups (cohorts) 2 to 6. The 'staircase' pattern for parameters 2 to 6, and 7 to 10 shows that we're allowing survival to vary among release cohorts as a function of age: in the first period following marking (*newborns*, parameters 2 to 6), and subsequent intervals (*mature*, 7 to 10). Note that in cohort 5, there are no 'mature' individuals.

Now, all that is left to do is to run the model, and add the results to the browser. All you need to do is remember that parameter 1 is fixed to 1.0. Go ahead and run the model, after first fixing the appropriate parameter to 1.0 – add the results to the browser – call the model '`S(a2 - cohort) - PIM`' (we add the word PIM to indicate the model was built by modifying the PIMs).



Model	AICc	Delta AICc	AICc Weight	Model Likelihood	No. Par.	Deviance
{S(a2-cohort) - PIM}	15143.6584	0.0000	1.00000	1.0000	9	3.6876

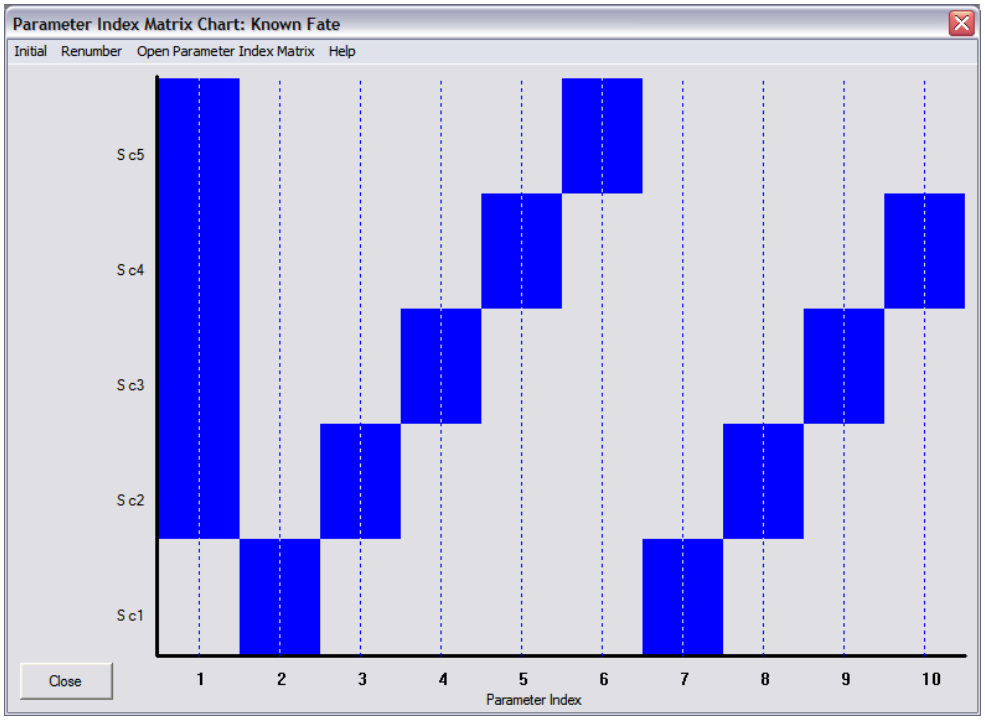
OK, what about the second model – model `S(a2)` (no cohort effect)? Well, if you reached this point in the book (i.e., have worked through the preceding chapters), you might realize that this model corresponds to

1	2	2	2	2
	1	2	2	2
		1	2	2
			1	2
				1

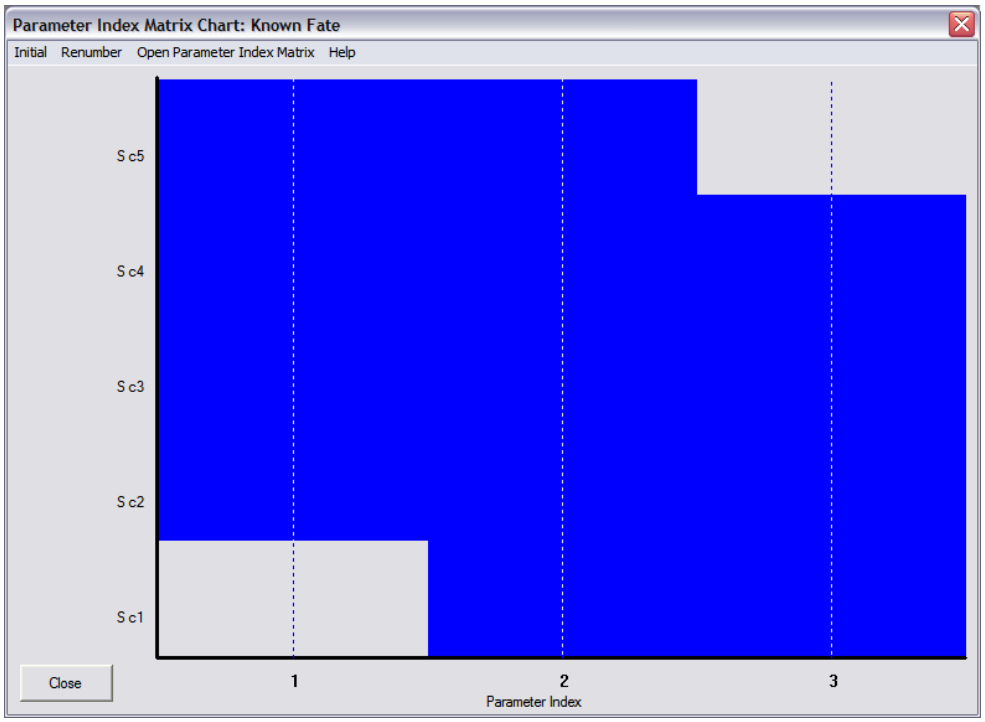
Again, if we add a parameter 1 to indicate the '`00`' cells preceding the first encounter within each cohort, and subsequently increment the parameter indexing for all other parameters by 1, we get

2	3	3	3	3
1	2	3	3	3
1	1	2	3	3
1	1	1	2	3
1	1	1	1	2

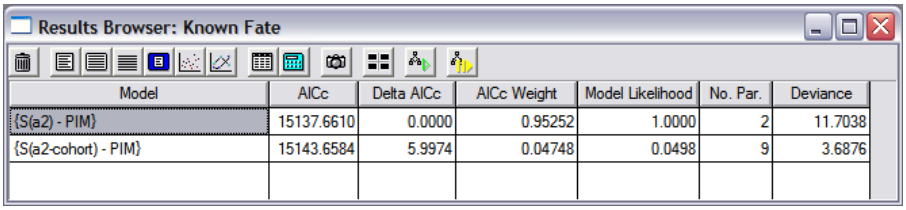
We can build this model conveniently by simply modifying the PIM chart for the preceding model `S(a2 - cohort)`. Recall that the PIM chart for that model was (see top of the next page)



So, to build model `S(a2)`, all we need to do is 'remove' the cohort variation for parameters 2 to 6, and 7 to 10 – this is shown in the modified PIM chart, below:



Now, run this model, first fixing parameter 1 to 1.0, label it '`S(a2) - PIM`', and add the results to the browser:



Model	AICc	Delta AICc	AICc Weight	Model Likelihood	No. Par.	Deviance
{S(a2) - PIM}	15137.6610	0.0000	0.95252	1.0000	2	11.7038
{S(a2-cohort) - PIM}	15143.6584	5.9974	0.04748	0.0498	9	3.6876

As expected, model `S(a2)` (the true, underlying model which we used to generate the data) gets virtually all of the AIC weight, relative to the other model. And, the reconstituted parameter estimates are very close to the true underlying values.

Now, while 'fiddling' with the PIM chart (and the underlying PIMs) is convenient for these simple models, we know from earlier chapters that there are structural limits to the types of models we can construct this way. Most obviously, we can't use the PIM approach to build models with additive effect. Ultimately, it's to our advantage to build models using the design matrix (DM), since all reduced parameter models can be constructed simply by manipulating the structure of the DM for the most general model. Let's build the DM for model '`S(a2 - cohort)`', which is the most general model of the two models in our candidate model set).

First, we start by writing out the conceptual structure of the linear model corresponding to this model:

$$S = \text{cohort} + \text{age} + \text{cohort.age}$$

The first term is fairly straightforward – we have 5 cohorts, so we need $(5 - 1) = 4$ columns to code for cohort. What about age? Well, look again at the PIM for this model:

2	7	7	7	7
1	3	8	8	8
1	1	4	9	9
1	1	1	5	10
1	1	1	1	6

Remembering that parameter 1 is fixed at 1.0, and is thus a constant. We can ignore it for the moment (although we do need to account for it in the DM). Pay close attention to the parameters along and above the diagonal. These represent each of the two age classes in our model – the vary among rows within an age class, but are constant among columns within a row, specifying cohort variation for a given age class, but no time variation (recall from Chapter 7 that a fully age-, time- and cohort-dependent model is generally not identifiable, since the terms are collinear). So, we have 2 age classes, meaning we need $(2 - 1) = 1$ column to code for `age`. What about `cohort`? Well, 5 cohorts, so $(5 - 1) = 4$ columns to code for `cohort`. Again, hopefully familiar territory. If not, go back and re-read Chapter 6.

But, what about the interaction terms (`age.cohort`) – do we need $(4 \times 1) = 4$ columns? If you think back to some of the models we constructed in Chapter 7 (age and cohort models), especially those models involving individuals marked as young only you might see how we have to handle interaction terms for this model. Recall from Chapter 7 that the interaction columns in the DM reflected 'plausible' interactions – if a specific interaction of (say) age and time wasn't possible, then there was no column in

the DM for that interaction. For example, for an analysis of individuals marked as young, there can be no interaction of age (young or adult) with time in the first interval, since if the sample are all marked as young, then there are no marked adults in the first interval to form the interaction (i.e., there can be no plausible interaction of age and cohort in the first interval, since only one of the two age classes is present in the first interval).

OK, so what does this have to do with our known-fate data? The key word is 'plausible' – we build interactions only for interactions that are plausible, given the structure of the analysis. In this case, there are only 2 true age classes (newborn, and mature). All of the other 'age' classes are 'logical' – we've 'created' them to handle the preceding '00' terms in the PIM. They are not true 'age' classes, since there are no marked animals in those classes. As such, there are no interactions between cohort and any of these logical '00' age classes – we need only consider the interactions of the two true 'biological' age classes (newborn, and mature), with cohort. But, how many columns? Look closely again at the PIM:

2	7	7	7	7
1	3	8	8	8
1	1	4	9	9
1	1	1	5	10
1	1	1	1	6

Pay particular attention to the fact that the 'newborn' age class shows up in all 5 cohorts, while the 'mature' age class shows up only in the first 4 cohorts (and not in the fifth). So, not all age.cohort interactions are 'plausible'. Which ones are 'plausible'? Well, both age classes are represented in the first 4 cohorts, but both age classes are represented only over intervals 2 to 4. Thus, we only need include cohorts 2, 3 and 4, in the interaction terms. See the pattern? If not, try again. It's very similar to problems we considered in Chapter 7.

OK, penultimate step – what about parameter 1? Well, as noted earlier, since it's fixed to 1.0, then it's simply a constant across cohorts, and thus, enters into the linear model as a single parameter.

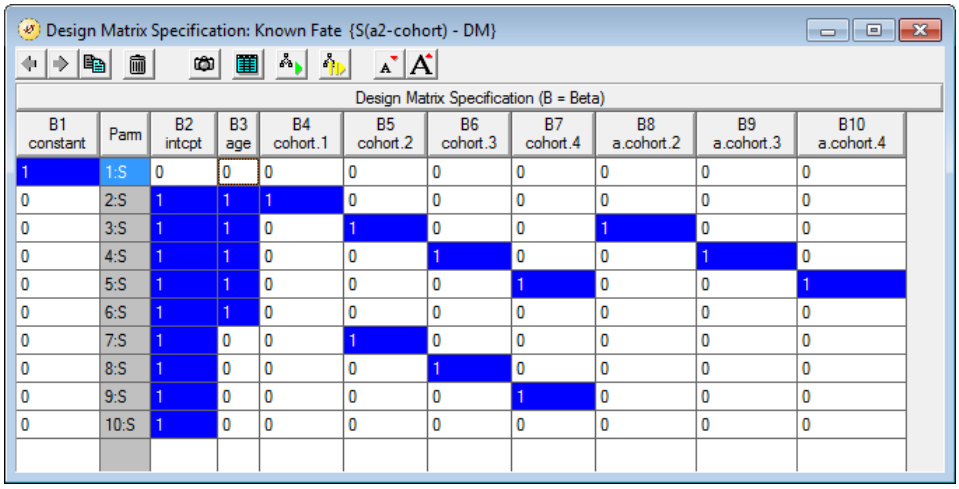
Now, finally, we're ready to write out the linear model corresponding to `S(a2 - cohort)`.

$$\begin{aligned}\hat{S} = {} & \beta_1(\texttt{constant}) \\ & + \beta_2(\texttt{intercept}) \\ & + \beta_3(\texttt{age}) \\ & + \beta_4(\texttt{c}_1) + \beta_5(\texttt{c}_2) + \beta_6(\texttt{c}_3) + \beta_7(\texttt{c}_4) \\ & + \beta_8(\texttt{age.c}_2) + \beta_9(\texttt{age.c}_3) + \beta_{10}(\texttt{age.c}_4)\end{aligned}$$

Is this correct? It has the same number of terms (10), as there are parameters in the PIM chart, so it would seem to be correct.

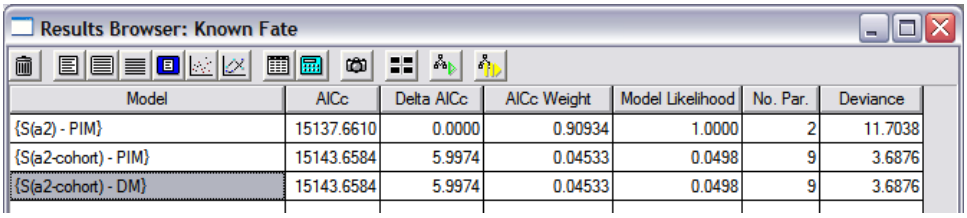
The next step, then, is to actually build the DM. We start by having **MARK** present us with a 10-column '**reduced**' DM as the starting point. The completed DM for this model is shown at the top of the next page. Column 1 (labeled `B1`) contains a single '1' - this represents parameter 1, which is a constant – fixed to 1.0 for all cohorts. The next column (labeled `B2`) represents the intercept for the 'age and cohort' part of the model. Column `B3` codes for `age` – `1` for newborn individuals, and `0` for mature individuals (note the different number of rows for each age class – this is key – 5 rows for newborns, and 4 rows for mature individuals). Columns `B4` to `B7` code for `cohort`. Note how the first row for newborn individuals for cohort 1 is coded, and note that this row does not show up for mature individuals – since, in cohort 1, there are no mature individuals! Finally, the interaction terms – columns `B8` to `B10`,

for those 'age.cohort' combinations that represent 'plausible' interactions.



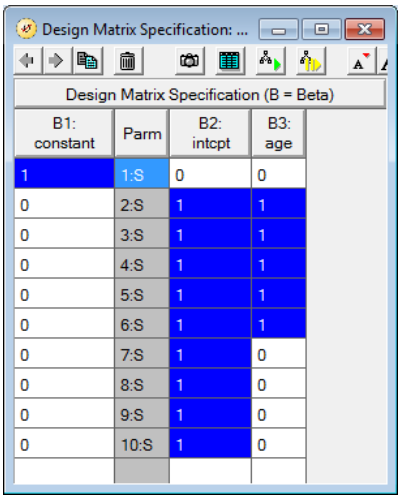
B1 constant	Parm	B2 intcpt	B3 age	B4 cohort.1	B5 cohort.2	B6 cohort.3	B7 cohort.4	B8 a.cohort.2	B9 a.cohort.3	B10 a.cohort.4
1	1:S	0	0	0	0	0	0	0	0	0
0	2:S	1	1	1	0	0	0	0	0	0
0	3:S	1	1	0	1	0	0	1	0	0
0	4:S	1	1	0	0	1	0	0	1	0
0	5:S	1	1	0	0	0	1	0	0	1
0	6:S	1	1	0	0	0	0	0	0	0
0	7:S	1	0	0	1	0	0	0	0	0
0	8:S	1	0	0	0	1	0	0	0	0
0	9:S	1	0	0	0	0	1	0	0	0
0	10:S	1	0	0	0	0	0	0	0	0

Go ahead and run this DM-based model (label it `S(a2-cohort - DM)`), and confirm that the results exactly match those for the model you constructed using the PIM chart, as shown below:



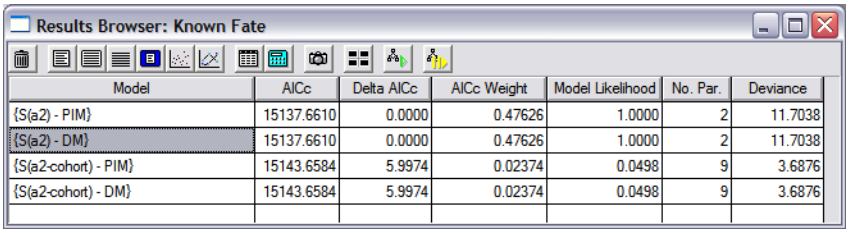
Model	AICc	Delta AICc	AICc Weight	Model Likelihood	No. Par.	Deviance
{S(a2) - PIM}	15137.6610	0.0000	0.90934	1.0000	2	11.7038
{S(a2-cohort) - PIM}	15143.6584	5.9974	0.04533	0.0498	9	3.6876
{S(a2-cohort) - DM}	15143.6584	5.9974	0.04533	0.0498	9	3.6876

Now that you have the DM for the general model, try constructing model `S(a2)` – the second model. We already did this a few pages back using the PIM approach, but we can generate the same model easily using the DM approach by simply deleting (i) the columns of the DM coding for `cohort`, and (ii) the (`age.cohort`) interaction columns:



B1: constant	Parm	B2: intcpt	B3: age
1	1:S	0	0
0	2:S	1	1
0	3:S	1	1
0	4:S	1	1
0	5:S	1	1
0	6:S	1	1
0	7:S	1	0
0	8:S	1	0
0	9:S	1	0
0	10:S	1	0

If you run this model, again you'll see the results exactly match those for model `S(a2)` built using the PIM approach:



Results Browser: Known Fate

Model	AICc	Delta AICc	AICc Weight	Model Likelihood	No. Par.	Deviance
{S(a2) - PIM}	15137.6610	0.0000	0.47626	1.0000	2	11.7038
{S(a2) - DM}	15137.6610	0.0000	0.47626	1.0000	2	11.7038
{S(a2-cohort) - PIM}	15143.6584	5.9974	0.02374	0.0498	9	3.6876
{S(a2-cohort) - DM}	15143.6584	5.9974	0.02374	0.0498	9	3.6876

We'll leave building an additional model `S(a2+cohort)` (i.e., a model with additive effects between age and cohort) to you as an exercise (hint: simply delete the interaction columns from the design matrix for model `S(a2*cohort)`).

So, we see that by treating different release cohorts as 'groups', we can use the known fate data type in **MARK** to handle staggered entry designs. Are there any other design types we can handle using known fate data? In fact, there are, but they involve using a different approach, based on treating known fate data in a live-encounter, dead-recovery context.

16.6. Known fate and joint live-dead encounter models

As noted earlier, the encounter history format for known-fate data is structurally similar to the classic `LDLD` format used for Burnham's live encounter-dead recovery analysis (Chapter 9). Recall that in that case, it is possible to observe an individual alive at the start of a particular interval (`L`), and dead at some point during the interval (`D`).

With a little thought, you might think that you could apply the live encounter-dead recovery model structure directly to known-fate data, if you simply fix the 'detection parameters' (r and p), and the 'fidelity parameter ' (F) to 1 (remember, for a known-fate data, we assume we know the fate of all individuals). With a little more thought, however, you might realize there is a complication – the live encounter-dead recovery model does not correctly handle the censoring of '`00`' `LD` pairs in a known-fate data. In the live encounter-dead recovery data type, the '`00`' is handled as an animal that was not detected as either alive or dead on this occasion. In a known-fate data, the '`00`' indicates that the animal was censored from the study. The distinction is made clearer in the following table, where we contrast the probability expressions, and interpretations, of the encounter history '`100010`' under the known-fate, and live-dead encounter models, respectively.

model	*probability*	*interpretation*
known fate	$S_1 S_3$	tagged at occasion 1, censored for interval 2 (not detected, or removed for some reason), and re-inserted into the study at occasion 3.
live-dead	$S_1 F_1 S_2 (1-p_2) S_3 p_3$ $+S_1 F_1 S_2 (1-p_2)(1-S_3)(1-r_3)$	(i) tagged at occasion 1, stays in sample, survives to occasion 2 but not encountered, survives to occasion 3, where it is encountered alive, not shot; (ii) tagged at occasion 1, stays in sample, survives to occasion 2 but not encountered, survives to occasion 3, where it is encountered alive, shot, but not recovered.

Clearly, the probability expressions differ considerably between the two model types. And, as such, you can't simply apply the live-dead encounter model to known-fate data without somehow accounting for the difference in how the '00' values in the encounter history are handled. Specifically, how can you 'tell' the live-dead model that a '00' means 'censored' and not either 'dead and missed', or 'live and missed'?

One way to handle this is to break up the encounter history and use a '-1 coding' – in other words, take the '10 00 10' encounter history and make it into 2 encounter histories as:

```
10 00 00   -1;
00 00 10    1;
```

Now, the live-dead model correctly handles the pair of encounter histories to allow the animal to be in the sample for the first interval, and then be removed from the sample. The animal is then re-injected back into the sample for interval 3. If all the r and p parameters are fixed to 1, and you also fix F to 1, then you will get the identical estimates of survival from the live-dead and known fate approaches.

To see that the preceding statement is true, first examine the probability of the first encounter history: $S_1 + (1 - S_1)(1 - r_1)$, which reduces to just S_1 because $r_1 = 1$. The probability of the second encounter history is $S_3 + (1 - S_3)(1 - r_3)$, which again reduces to just S_3. So, the product of these 2 encounter histories is identical to the probability of the original encounter history under the known fate model.

To make this 'trick' of splitting known fate encounter histories to allow censoring, let's consider a bit more complex example. Take the encounter history '10 10 00 10 11'. The known fate probability is just $S_1 S_2 S_4 (1 - S_5)$. The split encounter history for live-dead coding looks like:

```
10 10 00 00 00  -1;
00 00 00 10 11   1;
```

The probability expression corresponding to the first piece is just $S_1 F_1 p_2 (S_2 + (1-S_2)(1-r_2))$, which reduces to just $S_1 S_2$ because all F, p, and r parameters are fixed to 1. The second probability is $S_4 F_4 p_5 (1-S_5) r_5$, which reduces to $S_4 (1-S_5)$. The preceding might seem like a lot of work just to 'trick' the Burnham live-dead model into being able to handle known-fate data. Clearly, for 'typical' known-fate data, use of the known-fate data type in **MARK** is decidedly more straightforward (and, not surprisingly, why it's there in the first place). However, there are some situations where using the live-dead model is particularly helpful – we consider two such applications in the following.

16.6.1. Live-dead and known fate models (1) 'radio impact'

One of the most pressing questions with known fate data is 'What is the impact of the radio on the animal's survival?' A useful solution to this question can be obtained by marking some animals with non-intrusive tags. For example, one sample of ducks can be radio-marked, whereas a second can be banded with leg bands. Now, the data must be analyzed with a different model that incorporates the live detection probability p and the dead detection probability r.

The way to do this is to use the live-dead model, and specify 2 groups. The first group would consist of the radio-marked sample, where all the p, r, and F parameters are fixed to 1. The second group would consist of the leg-banded sample, where all the parameters are estimated. The power of this design comes into play when we compare a model with survival estimated separately for each group against the equivalent model but with survival estimated in common across both groups. The comparison of these 2 models provides a powerful test of the effects of the radios on survival. For a well-designed study, we might consider using a likelihood-ratio test between these 2 models to test the null hypothesis of no radio effect directly. Alternatively, we could use the Akaike weights to assign probabilities to which hypothesis we believe is most likely the truth.

16.6.2. Live-dead and known fate models: (2) 'temporary emigration'

The live-dead data type can also be used to estimate the fidelity (F) to a study area for known fate data. The approach is to code the `LD` pair as '`00`' for animals that leave the study area. That is, animals that leave the study area are not censored as if the radio failed, but rather included in the sample with `00` for periods when they are off the study area. Then, given that $p = 1$ and $r = 1$, F is estimated. So, consider what the probability would be for the encounter history '`10 10 10 00 00`' when $p = 1$ and $r = 1$ so that these terms are left out of the expression: $S_1F_1S_2F_2S_3(1-F_3)$. With F estimated, the only way to account for trailing `00` values is to have the animal emigrate. Remember that the Burnham joint live-dead data type assumes permanent emigration.

What if you want to model temporary emigration? The solution in this case is to use the Barker joint live-dead data type (see Chapter 9), where the parameter F' is the probability that an animal not available for capture (i.e., off the study area) returns to the study area. So consider the probability of the encounter history '`10 10 00 00 10`' with $p = 1$ and $r = 1$, along with no probability of sightings in between capture occasions (i.e., $R = 0$ and $R' = 0$): $S_1F_1S_2(1-F_2)S_3(1-F_3')S_4F_4'S_5$. The point here is that the Barker joint live-dead data type can also be used to estimate the temporary emigration probability from known fate data, and hence can also be used to assess the effects of radios on animals against a sample marked in a different fashion.

16.7. Censoring

Censoring appears 'innocent' but it is often not. If a substantial proportion of the animals do not have exactly known fates, it might be better to consider models that allow the sampling parameters to be < 1. In practice, one almost inevitably will lose track of some animals. Reasons for uncertainty about an animal's fate include radio transmitters that fail (this may or may not be independent of mortality) or animals that leave the study area. In such cases, the encounter histories must be coded correctly to allow these animals to be censored. Censoring often require some judgment.

When an animal is not detected at the end of an interval (i.e., immediately before occasion j) or at the beginning of the next interval (i.e., immediately after occasion $j+1$), then its fate is unknown and must

be entered as a '00' in the encounter history matrix. Generally, this results in 2 pairs with a '00' history; this is caused by the fact that interval j is a 00 because the ending fate was not known and the fact that the beginning fate for the next interval ($j + 1$) was not known. Censored intervals almost always occur in runs of two or more (e.g., '00 00' or '00 00 00'). See the example above where the history was '10 00 00 11'.

In this example, the animal was censored but re-encountered at the beginning of interval 4 (alive) and it died during that interval. It might seem intuitive to infer that the animal was alive and, thus, fill in the 2 censored intervals with '10 10' – this is incorrect and results in bias. The reason for this bias is because a dead animal is less likely to be encountered at a later occasion than if it lives. So, you have a biased sampling process – animals are mostly encountered because they are alive, and hence estimates of survival become too high if the '00' values are replaced with '10'.

Censoring is assumed to be independent of the fate of the animal; this is an important assumption. If, for example, radio failure is due to mortality, bias will result in estimators of $\hat{S}_i$. Of course, censoring reduces sample size, so there is a trade-off here. If many animals must be censored, then the possible dependence of fates and censoring must be a concern. In such cases, you probably should be analyzing the data with the live encounters-dead recovery data type, and explicitly estimate the p and r parameters.

16.8. Goodness of fit and known fate models

Consider a model where all the parameters are specific to both the time-interval, as well as the cohort (i.e., year marked and released). This is a fully-saturated model where there are as many unknown parameters as there are cells. Note, the saturated model always fits the data perfectly (by definition and design). The concept of a saturated model is necessary in computing model deviance. As discussed earlier in Chapter 5, the deviance of model j in the candidate model set is defined as

$$\text{Deviance} = -2\ln(\mathcal{L}_j(\theta)) - [-2\ln(\mathcal{L}_{\text{saturated}}(\theta))]$$

Typically, for most data types, the saturated model contains many uninteresting parameters – its use is primarily heuristic, in allowing use to estimate the deviance of some less general model, relative to the saturated model.

Now, if sample size is large (i.e., there are no cells with small expectations), then the deviance is asymptotically χ^2 with df equal to (the number of cells in the saturated model) - (the number of estimable parameters in model j). OK, fine, this is the basis of the likelihood ratio test discussed earlier in Chapter 5. What does this have to do with GOF testing for known-fate data?

Well, the problem with known-fate data is this – for known-fate models where all individuals enter at the same time (or even with staggered entry data), the saturated model where each cohort has its own survival estimate for each occasion is a sensible model, and as such, there is no way to estimate the deviance of the saturated model from itself. Because the saturated model fits the data perfectly, there is no GOF test for classical known-fate data. In reality, this is the same with all models in **MARK** – we just assume (i.e., make an assumption) that some reduction of the saturated model to a biologically reasonable model is okay, and use this reduction to assess GOF.

To help you understand this point, consider a simple radio-tracking study where 100 radios are put on a single age/sex class for one occasion. The saturated model is the simple survival estimate based on the binomial distribution. There is only one data point, hence one degree of freedom, and that df is used to make the estimate of survival. Thus, it is fairly obvious that there is no GOF test available – to obtain a GOF test, we would have to assume a reasonable biological model that is reduced from the saturated model. This selection can be pretty arbitrary (obviously).

16.9. Known-fate models and derived parameters

Typically you are doing a known fate analysis to be able to estimate survival over an interval, say 1 year. However, you also want to know something about how survival changes within the year, or maybe because of censoring and radio failure problems, you want to include animals in the analysis that only appeared for a period of time within the year period. For example, you are doing a bear study where you have staggered entries and some radio failures or collars that dropped off that you have kept track of on a monthly interval. However, you are interested in estimating annual survival. How do you get an estimate of annual survival from 12 monthly estimates?

MARK provides derived parameter estimates that are the product of all the estimates for the intervals in the PIMs. So, suppose you have a 3-year study, where you want 3 annual estimates of survival, but you have 36 months of data. The clever way of setting up your analysis is to define 3 groups for the known fate model, each with 12 occasions (months), with the 3 groups corresponding to the 3 years of interest. Then, when you examine the derived parameter estimates, you will find 3 estimates, representing the 3 years. Variances and covariances of the derived parameters are computed with the Delta method (Appendix B).

Derived parameter estimates can be used in model averaging and variance components analyses, so you further have all of the power of these methods available for your analysis of annual survival rates.

Part of the 'art' of how to set up the known fate data type is whether attribute variables should be incorporated as groups or individual covariates. Derived parameter estimates are a function of the individual covariates used to compute them, so whether age in the black duck example is treated as a group or an individual covariate won't make a difference in the estimates. However, if age is handled as a group variable, the derived estimates are clear. To get derived estimates when age is an individual covariate means that you must specify individual covariate values to obtain the correct estimates.

16.10. Known-fate analyses and 'nest success models'

Suppose you want/need to estimate the survival of radio-tracked animals when the animals are not monitored in discrete intervals, as generally required by the known fate data type. Consider that such data are no different than a set of (say) nests, where all the nests are not visited on the same day. As such, you could apply a 'nest success model' to the data – in such a model, the daily survival rate is estimated for each day of the study based on the sample of animals available on that day, and the exact day of death is not required (just as the exact day that a nest was destroyed is not known). We call these kinds of data '*ragged telemetry data*' because the sampling scheme is ragged, but useful estimates can still be obtained. Nest success analysis is the subject of our next chapter.

16.11. Summary

Known-fate models are a very important model type – most commonly applied in situations where individuals are marked with radios (i.e., radio telemetry studies). The presence of a radio makes is feasible (under usual circumstances) to determine the 'fate' of the individual: is it alive, or dead? Present, or absent? And so on. Although the assumption that detection and reporting probabilities are both 1.0 simplifies aspects of the modeling considerably, a number of complex, elegant approaches to handling known-fate data are possible – especially when known-fate data are combined with data from other sources.

CHAPTER 17

Nest survival models

Jay Rotella, *Montana State University*

In this chapter, we will introduce how to analyze nest survival data with program **MARK**. Nest survival is a key vital rate in the population dynamics of many birds. Accordingly, estimation of nest survival is a key aspect of many studies of breeding bird populations. Given the strong interest in nest survival, there is a rich literature detailing field techniques and estimation methods for this vital rate. The development of estimation techniques has been very active in recent years (e.g., see Dinsmore *et al*. 2002, Rotella *et al*. 2004, and references therein) and so most of the analyses described in this chapter are still relatively new.

At this point, you may be wondering why we need to go to the trouble of using program **MARK** for analyzing nest survival data. After all, with nests you know how many nests you found and you know the fate of every nest. Why not just compare the proportion of successful nests among groups with different attributes? Well, for starters (and many of you probably already know this), it turns out that this is not a valid approach for most data sets. As Harold Mayfield pointed out several decades ago (Mayfield 1961, 1975), such an analysis is only valid if destroyed nests can be found with the same probability as active ones. In most studies successful and unsuccessful nests are not found with equal probability, and most nests are found after egg laying has commenced. Mayfield pointed out that under these circumstances the proportion of successful nests, which he termed apparent nesting success, is biased high relative to actual nesting success, the proportion of nests that survive from initiation to completion. Klett *et al*. (1986:10) provide an excellent illustration of the source of this bias. In fact, it's so good that we'll repeat the essence of the example here.

Imagine a study in which only active nests can be found. Nests are found at various ages (i.e., days since the nest was started). Young leave the nest 35 days after nest initiation (in this example, immediately after hatching, as is typical for species with precocial young). Assume that the probability that a nest survives a single day (daily survival rate; DSR) is 0.96 regardless of calendar date or nest age, and thus, the true probability of a nest surviving from initiation to completion is 0.2396 (0.96^{35}). The following table (at the top of the next page) shows the bias involved in working with the proportion of successful nests if all nests aren't found on or before the day they're initiated. As you can see, overestimation is likely and can be severe.

© Cooch & White (2014)

age of nest when found	*probability of surviving to hatching*	*bias relative to actual hatching success*
0	$0.96^{35} = 0.24$	0.00
5	$0.96^{30} = 0.29$	+0.05
10	$0.96^{25} = 0.36$	+0.12
15	$0.96^{20} = 0.44$	+0.20
20	$0.96^{15} = 0.54$	+0.30
25	$0.96^{10} = 0.66$	+0.42
30	$0.96^{5} = 0.82$	+0.58
34	$0.96^{1} = 0.96$	+0.72

Mayfield developed an *ad hoc* estimator of nesting success that overcomes the bias associated with estimates of apparent nesting success. His approach calculates the daily survival probability for only the days that nests were under observation. Thus, the Mayfield model accounts for the fact that some nests are not under observation starting with the day of nest initiation.

To see how Mayfield's method works, consider the following example where 10 nests were found at an age of 0 days, another 10 were found at an age of 14 days, and another 10 were found at an age of 28 days. After being discovered, each nest's fate was checked every 7 days until it failed or reached an age of 35 days. The dataset can be summarized in the following table, which tabulates the number of nests in the sample that survived to a given age – for example, we see that among the 10 nests found at age 14 days, 8 survived to 21 days (i.e., an additional week), 7 survived to 28 days (i.e., an additional 2 weeks), and so on:

initial sample size	*age of nest when found*	*7 days*	*14 days*	*21 days*	*28 days*	*35 days*
10	0 days	7	6	5	3	3
10	14 days			8	7	4
10	28 days					8

The next step is to run Mayfield's calculations on the data. Key steps are to calculate (1) how many nests were unsuccessful and (2) how many days nests were exposed to potential nest failure (termed exposure days). The number of unsuccessful nests is easy and intuitive to calculate: (10 − 3) + (10 − 4) + (10 − 8) = 15 failed nests. Exposure days may be a bit less obvious, but here's how it's calculated. All nests in this example were checked every 7 days. So, if a nest survived the interval between 2 visits, it was exposed to potential nest failure on each of the 7 days between visits. Thus, each time a nest survived a re-visit interval, we simply need to add 7 days to the total number of exposure days. For nests found at an age of 0 days, nests survived a total of 24 7-day intervals (7+6+5+3+3), which yields 168 exposure days. For nests first found at an age of 14 days, nests survived a total of 19 7-day intervals (8 + 7 + 4), which yields 133 exposure days. For nests first found at an age of 28 days, nests survived a total of 8 7-day intervals, which yields 56 exposure days.

But, what to do with nests that failed between two nest visits? We don't know when the failure occurred and so don't know how many days of exposure to use. Mayfield calculations typically use 1/2

the interval length in the cases of failures that occurred on an unknown date between two nest visits. So, these calculations are pretty simple to make as well. We know that 15 nests failed and, because all intervals in this simple example were 7 days long, each failed nest is assumed to have survived through 3.5 exposure days (one half of the 7-day interval). Thus, we need to add 52.5 exposure days (15 nests x 3.5 days per nest) to the total. Using the values we've just calculated, we see that 15 nests failed during a total of 409.5 exposure days (total = 168 + 133 + 56 + 52.5). Thus, the Mayfield estimate of daily mortality probability for nests is (15/409.5) or 0.0366. The Mayfield estimate of daily survival probability is simply one minus the daily mortality rate, or (1 – 15/409.5) or (1 – 0.0366) or 0.963. Mayfield's *ad hoc* estimator thus focuses on daily survival probabilities and circumvents the problems inherent in using apparent nest survival probabilities.

Subsequent to Mayfield's (1961, 1975) publications, a variety of authors published a maximum-likelihood approach to estimating daily survival probabilities and nesting success for data from nests that were visited periodically (Johnson 1979, Hensler and Nichols 1981, Bart and Robson 1982). The method is based in statistical theory and provides estimates of the mean and variance of daily survival rate. Thus, use of the maximum-likelihood estimator (MLE) of daily survival probability and its variance is recommended over use of Mayfield's *ad hoc* estimator despite the fact that they produce very similar point estimates of daily survival (and are in fact the same estimator in the case where nests are visited every day).

Let's review some key points made in this chapter so far: we can't validly use the proportion of successful nests as a surrogate for true nest success so we want to use maximum likelihood estimates of DSR and its associated variance instead. As we've seen in earlier chapters, program **MARK** is a very useful tool for doing just this sort of thing. Once we have estimates of DSR, we can raise it to the appropriate power (number of days it takes to go from nest initiation to nest completion [e.g., 35 days]) to estimate true nest success, e.g., $0.96^{35} = 0.2396$. With estimates of the variance of DSR, we can also estimate the variance of nest success ($\text{var}(0.96^{35}) = [(35 \times 0.96^{34})^2 \times \text{var}(\text{DSR})]$. So, by this point we hope you're starting to see how program **MARK** might be useful. But, it gets better because program **MARK** allows one to evaluate a rich variety of competing models of nest survival rate and go well beyond what was possible with basic Mayfield estimation.

Next, we use data collected on Mallard (*Anas platyrhynchos*) nests in North Dakota to introduce the analysis of nest-survival date in program **MARK**. This data set is part of a larger data set collected and analyzed by Stephens (2003) and the same data set used by Rotella *et al*. (2004) in their example analysis. It contains information from 565 nests that were monitored on 18 sites during a 90-d nesting season. Nests of various ages were found during periodic nest searches conducted throughout the nesting season. Once a nest was found, it was re-visited every 4 to 6 days to determine its fate (a binary outcome). For each nest, several covariates of interest were measured: (1) the date the nest was found; (2) the nest's initiation date, which provides information about the age of the nest when it was found, its age during each day of the nesting season, and its expected hatch date (35 days after nest initiation, which is when young typically leave the nest in this species); (3) a measure of how much the vegetation around the nest site visually obscured the nest; (4) the proportion of grassland cover on the 10.4-km^2 study site that contained the nest; and (5-7) the habitat type in which the nest was located (3 indicator variables, each coded as `0` or `1`, that were used to distinguish among nests found in 4 different habitat types: native grassland, planted nesting cover, wetland vegetation, and roadside right-of-ways).

17.1. Competing models of daily survival rate

Up to this point, we haven't discussed competing models. We've simply focused on the idea that we need to estimate daily survival rate (DSR) and can use program **MARK** to do so. But, typically we're

interested in evaluating competing hypotheses about sources of variation in nest survival. DSR might vary spatially due to changes in features such as habitat characteristics and predator communities. DSR might also be hypothesized to vary temporally. For example, one might hypothesize for a given species that nests are more vulnerable later in the nesting season as predators develop a search image for nests and key into nests as food sources. Thus, DSR may be predicted to be relatively high early in the season, to drop off in the mid-season, and to stay low late in the season. One might also be interested in changes in DSR that are associated with changes in nest age. For a songbird that feeds young in the nest, one might hypothesize that DSR will be lower during the nestling stage, when parents make many foraging trips per day to and from the nest to provide for begging young, than it will be during egg-laying and incubation stages, when fewer cues are provided to predators. In a multi-year study, we might wonder if DSR differs among years.

It seems safe to say that we can think of lots of possible sources of variation in DSR. So, it also seems reasonable to say that we will want to be able build and evaluate competing models of DSR. This may seem readily obvious, but it was only recently that methods were developed for evaluating multiple-regression type models of DSR (Dinsmore *et al*. 2002; and, yes, these methods were first developed in program **MARK**).

Most studies of nest survival have used Mayfield's *ad hoc* estimator of DSR or the maximum likelihood estimator of Johnson (1979) or Bart and Robson (1982), and these methods assume that DSR is the same for all nests on all dates and for all nest ages (nest fates are independent and identically distributed: *iid*) within the sample. To attempt to meet the assumption of homogeneity of fates within a given sample, one can stratify data prior to analysis and create sets for which fates can reasonably be considered *iid*. For example, data can be analyzed separately for different nesting stages, habitats, years, species, or combinations of these. However, such stratification can be limiting and may prevent researchers from exploring the full suite of models that are likely of interest. This is especially difficult when some of the multiple factors that are thought to affect DSR are measured on continuous rather than categorical scales. Although one can certainly discretize continuous variables and convert them to categorical values, this won't always be desirable.

Fortunately, program **MARK** allows us to model DSR as a function of multiple covariates and to compare competing models. But, before we can begin building models, we need to consider the data input for nest-survival data. It turns out that there are a few unique features to this data type.

17.2. Encounter histories format

Minimally, four pieces of information are required for each nest: (1) the day of the nesting season on which the nest was found; (2) the last day the nest was checked when alive; (3) the last day the nest was checked; and (4) the fate of the nest (0 = successful, 1 = depredated). Program **MARK** uses these key pieces of information to generate an encounter history for each nest in live/dead (**LDLD**) format (see Chapter 2). Although you'll never see the **LDLD** type encounter histories in program **MARK** for this data type, we will explore them a bit more below as they can help us understand what's taking place with nest survival data.

The days mentioned above refer to standardized days within the study's nesting season. These standardized days are obtained by first calculating the earliest date on which you began recording data on nest survival, treating this date as day 1 of the nesting season, and then sequentially numbering all subsequent days for which you observed nests up to the last date of data collection. For the Mallard data set used in this example, data collection began on 24 April and continued for 90 days. So, 24 April was standardized as day 1 and the last day was day 90.

The next step is to create an encounter history for each nest based on the critical information described above. In the mallard study, the first nest was found on day 1 (April 24) of the season with 1 egg. This nest was observed periodically over the next 35 days and was successful. Thus, this nest had the following encounter history:

```
/* 1  */ 1 35 35  0 1;
```

Here, the encounter history begins with a comment (`/* 1 */`) referring to the nest's identification number (this is quite handy when data checking but not essential to provide). The key information follows the comment: (1) first is the day the nest(s) was/were found, labeled time *i*; (2) next is the last day the nest(s) was/were known to be present, labeled time *j*, which for successful nests should be the day the nest attempt was successfully completed (as it is in this example); (3) then comes the last day that the nest(s) was/were checked or, for successful nests, the day that the nest attempt should have been successfully completed (as it is in this example), labeled time *k*; (4) next is the fate of the nest(s): 0 means successful, and 1 means destroyed or unsuccessful; and (5) the last value is the number (frequency) of nests that had this history (this will usually be 1 but doesn't need to be if multiple nests have the same encounter history). Later in the study, a nest had this encounter history:

```
/* 1931  */ 63 81 85 1 1;
```

From this, we can see that this nest was found on day 63 of the nesting season, last known to be active on day 81, and last checked on day 85. The nest was unsuccessful and was destroyed sometime between day 81 and day 85.

You may have noticed that for successful nests, all the information about survival time is contained in times *i* and *j*. If you did, you're paying attention and you're right. Time *k* is only used for unsuccessful nests to bracket the interval when the nest was destroyed. Another point worth emphasizing is the fact that dates *j* and *k* should not extend beyond the nest's successful completion date even if the actual nest check was done after the hatch date (or, in the case of nidicolous species with a nestling stage, beyond the expected fledging date). If you put in the actual check dates rather than completion dates, you'll give the nest credit for surviving days when it was no longer subject to possible failure — definitely not something we want to do.

The histories above could be extended to include individual covariates after the 5 required variables. In the mallard example, the following covariates were added to the encounter histories: (1) a measure of how much the vegetation around the nest site visually obscured the nest; (2) the proportion of grassland cover on the 10.4-km^2 study site that contained the nest; (3) an indicator variable that was coded as **1** if the nest was in native grassland and **0** otherwise; (4) an indicator variable that was coded as **1** if the nest was in planted nesting cover and **0** otherwise; (5) an indicator variable that was coded as **1** if the nest was in wetland vegetation and **0** otherwise; and (6) the age of the nest on the first day of the nesting season. Because the age variable is for the first day of the nesting season, this value will be negative for all the nests in the sample except those that were initiated (started) on or before day 1.

Several items are noteworthy about the covariates. First, date, which may be of interest as a covariate in some models, is incorporated into the required fields of the encounter history, and so, there is no need to include date information elsewhere. One might, of course, wish to have a variable(s) for year of study in a multiple-year study. Second, in this example, there were 4 habitat types of interest but only 3 indicator variables are included: the 4th habitat type (roadside right-of-ways) is indicated if all 3 of the indicator variables have a value of **0**. Third, the age variable as described above might seem a bit odd at first glance. Why do we want to work with the age of each nest on the first date of the nesting season? The short answer is it allows us to generate each nest's age on every other day of the nesting season and

these are critical to have if we think that DSR might vary by nest age. A longer answer comes below when we use the age variable in modeling.

Is that it for covariates? Well, no. As we've seen for some other data types, the encounter histories are assigned to groups, so we could also group the data in various ways, which is another way of incorporating a group covariate. Group covariates can also be incorporated through the design matrix in ways that will be explained below.

In the mallard example, data were originally recorded in interval-specific form, i.e., each row of data contained information for one observation interval for an individual nest. For the 2nd nest found in the study, the following rows of raw data were recorded:

Nest ID	Species	Study Site	Habitat Code	Obs	t	IFate	SDate	Sage	Robel	PpnGR
2	MALL	14	PICov	1	5	0	1	3	0.88	0.96
2	MALL	14	PICov	2	5	0	6	8	0.88	0.96
2	MALL	14	PICOv	3	4	0	11	13	0.88	0.96
2	MALL	14	PICov	4	6	1	15	17	0.88	0.96

Here the columns represent (1) the nest's identification number, (2) a species code, (3) the study site, (4) a habitat code, (5) a number indicating which observation interval is represented on this row for this nest, (6) the number of days in the observation interval, (7) the nest's fate for this observation interval (0 = successful, 1 = failed), (8) the day of the nesting season at the start of the interval, (9) the age of the nest at the start of the interval, (10) a measure of how much the vegetation around the nest site visually obscured the nest, and (11) the proportion of grassland cover on the 10.4 km^2 study site that contained the nest. The data for nest #2 were re-formatted for analyses in **MARK** as:

```
/* 2  */ 1 15 21 1 1 0.88 0.96 0 1 0 3;
```

To create the actual input file for **MARK**, the data for each nest to be used in the analyses must be formatted appropriately. In addition, the first non-comment line of the file needs to consist of the statement '`Nest Survival Group=1 ;`'. Following this command are the data for all of the nests in the 1st group. When this is done for the mallard data, the first portion of the encounter history file appears as follows:

```
 Nest Survival Group=1;

/* 1 */  1  35  35  0  1  4.500  0.9600  0  1  0  1;
/* 2 */  1  15  21  1  1  0.875  0.9616  0  1  0  3;
/* 4 */  1  11  15  1  1  2.750  0.9616  0  1  0  3;
/* 5 */  1  31  33  1  1  3.375  0.9616  0  1  0  3;
/* 6 */  2   2   7  1  1  1.875  0.9616  0  0  0  3;
/* 7 */  2   7  12  1  1  2.750  0.9616  1  0  0  4;
```

Note: The **MARK** help file contains much useful information about how to build the encounter history file. It is definitely worth taking the time to review the information provided.

17.3. Nest survival, encounter histories, & cell probabilities

In **MARK**, the data type for the encounter history is `LDLD` (although you won't code it this way in the `INP` file - **MARK** will handle this for you). The cell probability is modeled as the product of the survival probabilities from time i to time $j-1$. For successful nests, i.e., fate=0, the last time seen alive (j) and the last time (k) should always be the same and equal to the day the nest attempt was completed (which may be hatch date for species with precocial young, or fledging date for species with altricial young). For unsuccessful nests, fate> 0, $j < k$.

For successful nests, time k is ignored. For situations where the date of hatch is not known exactly, none of the times should exceed the potential nest completion date, because the nest should not be considered at risk of failure when a nesting attempt has already been completed. For unsuccessful nests, the cell probability for the nest is taken as the product of the survival rates from time i to time $j-1$, times 1 minus the product of survival from j to $k-1$.

The following series of examples for 5 occasions will clarify how to code nest survival data with the triplet i, j, and k.

triplet	*history and interpretation*
$i=1, j=3, k=5, fate=1$	`1010100001`, with cell probability $S_1S_2(1-S_3S_4)$
$i=1, j=3, k=3, fate=0$	`1010100000`, with cell probability S_1S_2
$i=1, j=3, k=3, fate=1$	invalid, because the nest was observed both present and destroyed on day 3
$i=1, j=1, k=3, fate=1$	`1000010000`, with cell probability $1-S_1S_2$
$i=1, j=1, k=3, fate=0$	invalid because the nest was observed present only on day $i=1$. If the nest was still present on day 3, then j should have been coded as $j=3$
$i=1, j=3, k=5, fate=0$	partially invalid because the nest was observed for the interval 1 to 5, when the nest was successful, but the coding shown will only use the data from 1 to 3, giving `1000100000`, with cell probability S_1S_2. For $fate=0, j=k$
$i=3, j=3, k=3, fate=0\ or\ 1$	invalid because the nest was not observed over an interval for either fate

The key difference between *known-fate* (discussed in the preceding chapter) and *nest survival* data types is that with nest survival data, we don't know exactly what day the nest was destroyed for cases where the nest was unsuccessful. Thus, consider the cell probability for the following:

```
/*GGOO, 1995-076*/    53  59  63  1    1 4;
```

This nest was found on day $i=53$, checked and found still present on day $j=59$, and found destroyed (i.e., failed) on day $k=63$. The last variable (an individual covariate) in this example is the age of the nest at the time it was first found – 4 days old for the first nest.

The cell probability corresponding to this nest would look like:

$$S_{53}S_{54}S_{55}S_{56}S_{57}S_{58}\,(1-S_{59}S_{60}S_{61}S_{62})$$

The first portion of the expression models the survival of the nest from day $i = 53$ to day $j = 59$. The second portion of the expression, in brackets, models the failure of the nest during the interval from day $j = 59$ to day $k = 63$. That is, had the nest been successful during this interval, the probability would have been $S_{59}S_{60}S_{61}S_{62}$. Because the nest was destroyed at some time during that interval, this quantity is subtracted from 1. The complete encounter history for this example would consist of 52 pairs of '00', followed by the following string with blanks inserted to delimit occasions denoted on the second line:

```
10 00 00 00 00 00 10 00 00 01

53 54 55 56 57 58 59 60 61 62
```

Needless to say, having to write this out in full on your own would be a rather tedious exercise. Fortunately, **MARK** handles all of this for you 'behind the scenes'. But, it is important to understand how the probability statements are created.

begin sidebar

Effective sample size and nest survival analyses

The effective sample size for AIC_c is computed somewhat differently for the nest survival model than other models in **MARK**. Typically, each binomial trial in a model provides one degree of freedom. For the nest survival model, each day that the nest is known to survive contributes 1 degree of freedom because each day is a binomial trial where the result is known. However, the interval in which a nest fails only contributes 1 degree of freedom also, because the exact day of failure is not known, but only that the nest failed during the interval.

Thus, a record like

```
/*YGBB, 1995-074*/          44  55  55  0    1  18};
```

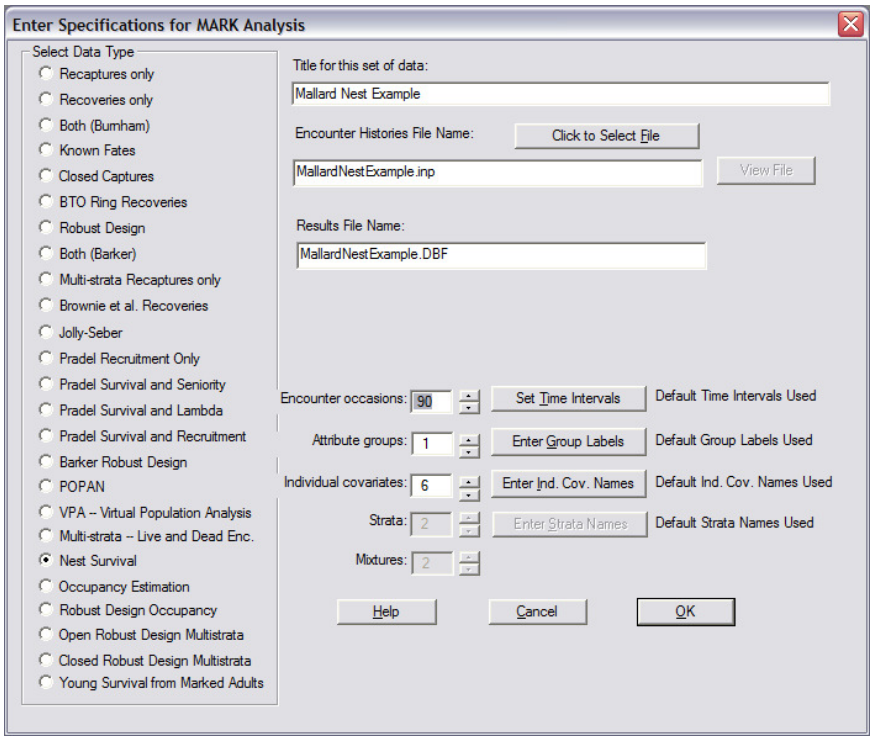
contributes 10 degrees of freedom, because there are 10 binomial trials where the nest was known to succeed. However, the following record only contributes a single degree of freedom, because the nest failed somewhere in the interval 58-61.

```
/*WGGW, 1995-078*/      58  58  61  1    1  18;
```

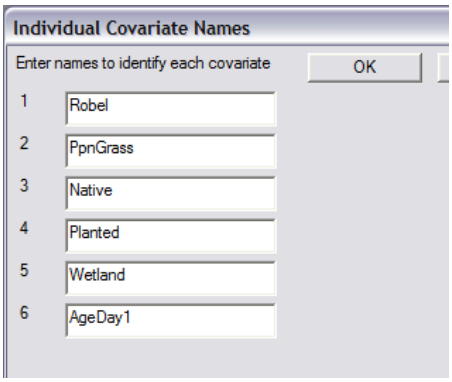
end sidebar

17.4. Building models

OK, this is what you've probably been patiently waiting to get to all along so finally, here is some information on how to actually build competing models for nest survival data in **MARK**. Before we can get started, we need to start **MARK** and opt to create a new **MARK** database file from your nest-survival encounter history file using the '**File**' menu and the '**New**' option. From this point, choose the '**Nest Survival**' data type and fill in the required information. For our mallard example, we have a 90-occasion study (data were collected across 90 days), 1 attribute group, and 6 individual covariates.



If you'd like, you can enter names for the individual covariates. As we have 6 that we'll use in the model-building exercises below, it's probably best to name them. So, choose to '**`Enter Individual Covariate Names`**' and enter the following: `Robel`, `PpnGrass`, `Native`, `Planted`, `Wetland`, and `AgeDay1`.



Model 1: Constant Daily Survival Rate

Now that we have created a new database, we'll start our model building with the simplest model, the maximum likelihood version of the Mayfield model. Remember, in this model, we assume that all nests in the sample under consideration have the same DSR on every day. So, we simply need to constrain DSR accordingly.

This can be done with the PIM chart, the survival PIM window, or by choosing to run the appropriate pre-defined model, which is listed appropriately as the `S(.)` model (although, again - as noted before, we do not favor relying on pre-defined models - they're available in **MARK** to expedite analysis of

common models, and should only be used if you have a firm grasp on what you're doing).

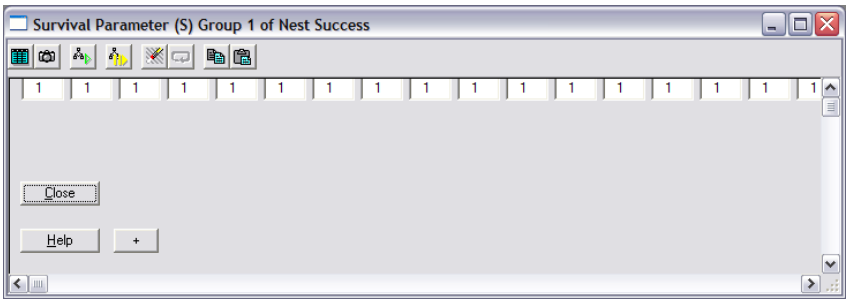
begin sidebar

PIMs and nest survival

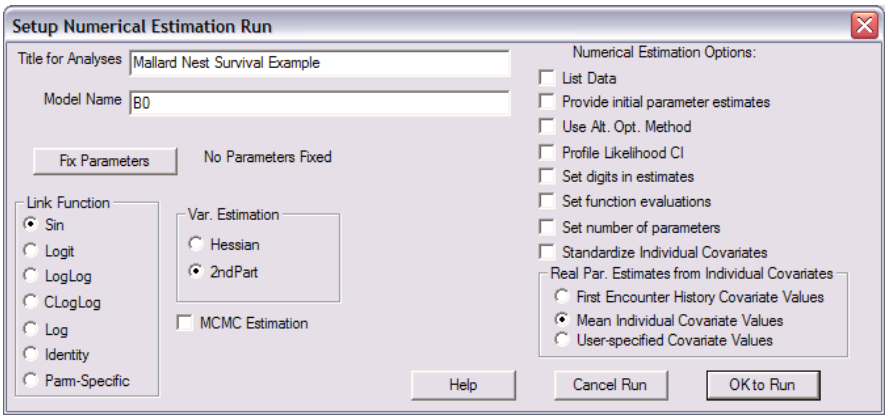
This brings up a question: how many PIMs are there for this nest survival dataset? Well, it turns out that there is only one. That's right: we've only got one group, and there are no parameters other than survival (DSR) to estimate for this data type. It also brings up the question of how many cells there are in the PIM. For this dataset, there are 90 encounter occasions and thus, there are 89 cells in the PIM to estimate the survival probabilities across the 89 intervals. Unfortunately, for this example, the PIM is so wide we can't show you the whole thing - but, try it yourself.

end sidebar

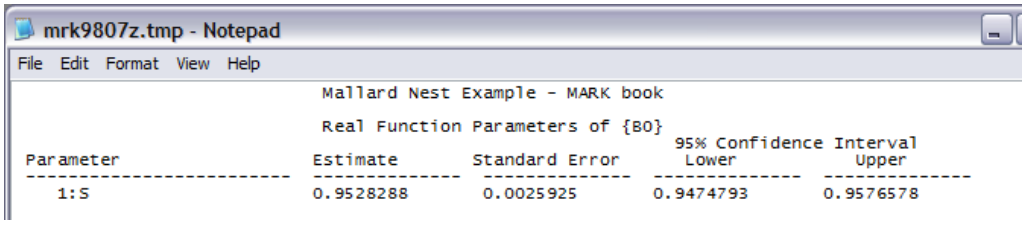
OK, back to model building. To set up this model using the PIM window, simply set all the values in the PIM to 1, which, of course, is the $S(.)$ model we're after here. The easy way to set all the values to 1 is to right-click the PIM window, and select '**`Constant`**'.



Next, go ahead and run this model. You might name the model something like '`Constant DSR`' or '`B0`' (as it is an intercept-only model, and we usually denote the intercept as β_0 or β_1 (depending on your convention), or '`B0`' or '`B1`'in ASCII, respectively. Here we will use β_0 for the intercept.). Be sure to turn off the option '**`Standardize Individual Covariates`**', because this option is not needed for these data (i.e., the covariate values are scaled to not produce numerical problems), and because we will not want to scale the `AgeDay1` variable in a future analysis. So, to maintain compatibility of the models, we uncheck the '**`Standardize Individual Covariates`**' box.



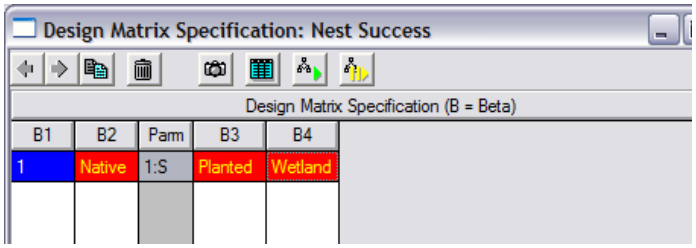
Once the model has run, you can examine the real parameter estimates whereupon you'll see the following:



This is the estimate of DSR when DSR is constrained to be constant across all nests and all dates in the sample. For mallards, it is typically considered that a nest must survive 35 days in order to make it from the 1st day of egg laying to nest completion, which is the day the young leave the nest. So, you could raise the estimated DSR to the 35th power to obtain a point estimate of nest success, which is 0.184.

Model 2: DSR Varies by Habitat Type

Often there is interest in whether or not DSR varies among nests found in different habitat settings. Although there are certainly many metrics that can be used to describe the habitat conditions associated with a nest, researchers often categorize habitat conditions and use a metric such as habitat type. In the study of mallard nests, the researchers used 4 habitat types, which you'll remember were described using 3 dummy variables as individual covariates. To build a model that allows DSR to vary by habitat type, you'll need to use the design matrix and the individual covariates that tell **MARK** which habitat type each nest was in. The following design matrix can be used.



begin sidebar

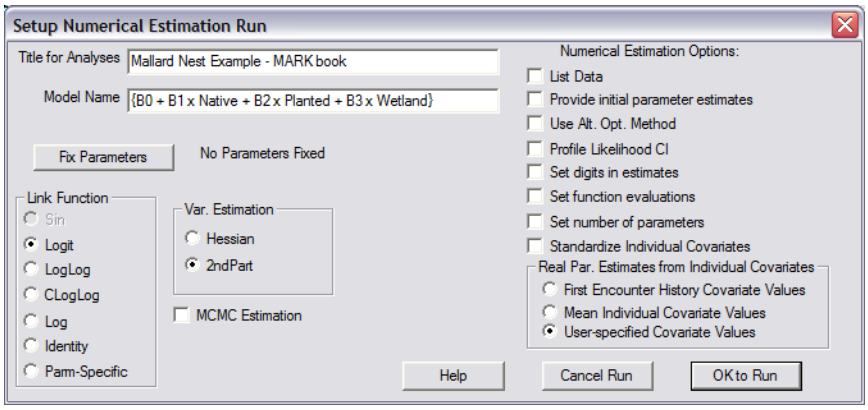
DM, groups and nest survival

If you have experience using groups in **MARK**, you may recognize that encounter histories for nests in different habitats could be put in different groups. If this were done, you would have a different PIM for DSR for each group and could evaluate a model that allows DSR to vary among habitat types without using a design matrix. In this example, all of the data were entered in a single group, the habitat type associated with each nest is input as an individual covariate, and the design matrix is used to build models in which DSR is allowed to vary among habitat types. The modeling results obtained will be the same regardless of whether habitat type is input using different groups or individual covariates.

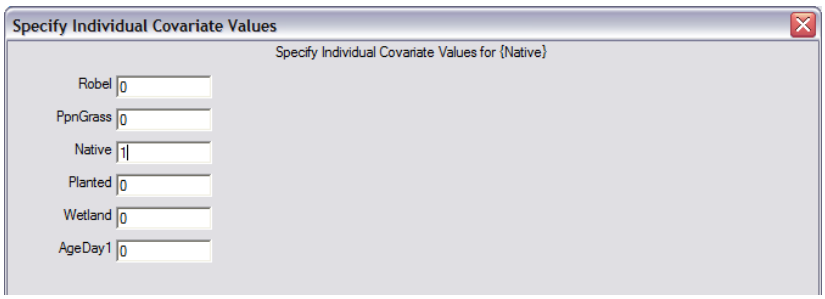
end sidebar

Once you've specified the design matrix appropriately, you're almost ready to run this model. When

you name the model, you might call it '`Habitat Type`' or, in keeping with the naming strategy used in the previous model, you might call it '`B0 + B1 x Native + B2 x Planted + B3 x Wetland`'. Once again, be sure to turn off the option 'Standardize Individual Covariates', because this option is not needed for these data. Next, you need to think through which habitat you'd like the estimates of the real parameters to be for. One easy way to control this is by specifying that you want to use '**`User-Specified Covariate Values`**' (check the button in the lower right corner of the setup window. For more on this topic, look in the **MARK** Help file under '**`Individual Covariates and Real Parameter Estimates`**.').



Let's assume for now that you want to get estimates for `Native` vegetation (we'll get estimates for other habitat types later). Specify that you want to estimate the real parameters when `Native` = 1, i.e., when the habitat type is native grassland. For this model, all other parameter values can be left as 0: values for `Robel`, `PpnGrass`, and `AgeDay1` are ignored as they're not in the model; values for `Planted` and `Wetland` need to be 0 to ensure we're only estimating DSR for nests in `Native`.



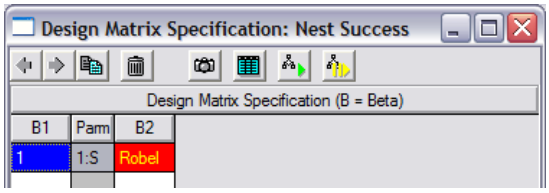
We want to know the habitat-specific estimates of DSR. To get those, we'll need to run the model 3 more times and choose to base the real parameter estimates from user-specified covariate values. When you run this model, you'll see that (1) the results for this model suggest that the simpler intercept-only model (our '`S(.)`' or '`B0`' model) is more appropriate and (2) the estimate of DSR for native grassland is 0.946 (SE = 0.005). Now, you can re-run the model 3 more times: setting `Planted=1` and all other variables to 0 on one run, setting `Wetland=1` and all other variables to 0 on another run, and then finally running the model with all variables set to 0. What will this last model run allow us to estimate? If you figured out that this will allow you to obtain an estimate for roadside habitat (i.e., the habitat that a nest is in for this example if it's not in planted cover, native cover, or a wetland), then you're getting this (or this is old hat to you and you might consider reading something else!).

Notice that you don't want multiple versions of this same model sitting in the Results Browser as they influence the model weights. But, you may want to look at each and see what the habitat-specific estimates of DSR are (`Native`: 0.946 [SE=0.005]; `Planted`: 0.956 [SE=0.003]; `Wetland`: 0.951 [SE=0.011]; and `Roadside`: 0.956 [SE=0.009]).

You now see how **MARK** can be used to evaluate basic models of DSR using maximum likelihood estimation and AIC. This alone would make **MARK** a very useful tool for analyzing nest survival data. However, we can do much more. To showcase a few of the other types of models that can be run, we'll now explore models that allow DSR to vary based on factors such as habitat metrics measured on a continuous scale.

Model 3: DSR varies with vegetation thickness (continuous covariate)

The input file contains an individual covariate labeled `Robel` (a continuous measure of how much the vegetation around the nest site visually obscured the nest). One way to evaluate whether DSR varies with `Robel` is to use the following design matrix:

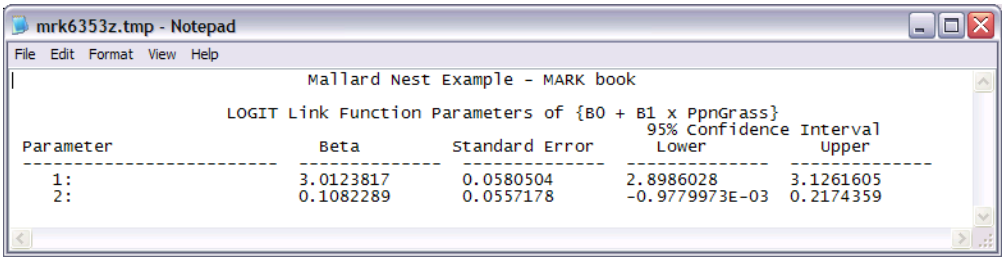


When this model is run (you might call it '`B0 + B1 x Robel`'), you find that this model receives less support than the simpler intercept-only model (our `S(.)` model).

Model 4: DSR varies with the amount of native vegetation in the surrounding area

The input file contains an individual covariate labeled `PpnGrass`, which you may recall is a continuous measure of the proportion of grassland cover on the 10.4 km^2 study site that contained the nest. Given interest in relationships between nest survival and spatial features of vegetation, models containing such covariates may be of interest in some studies. And, as you can probably now readily envision, such models are readily evaluated in **MARK**.

You can use the design matrix to build a model containing `PpnGrass` and if you do, you will find that this model receives more support than any of the models discussed so far. In fact, there is evidence that DSR is higher on sites that contain more grassland cover than on sites with less cover.



But, we'll assume that we have other *a priori* models that are of interest and will run them all before

getting into the details about inferences that can be drawn from the analysis.

Model 5: DSR varies across the nesting season

One might predict that nests will be more or less vulnerable as the season progresses. Changes might be predicted to occur because the abundances of predators and alternate prey are thought to vary seasonally or because vegetation characteristics may change dramatically over the course of the season in some habitat settings. Certainly other interesting hypotheses might exist as well. For now, let's just evaluate a model that constrains DSR across the nesting season such that it is forced to follow a trend (a linear trend on the logit scale).

To build a trend model, go ahead and open the PIM window and set the values to all different (i.e., so that the values in the 89 cells of the PIM take on the values $1, 2, 3, \ldots, 89$). Next, open a reduced design matrix with 2 columns. Now, because our PIM had 89 parameters, the design matrix has 89 rows and 2 columns. Fill the first column in with 1's (we want to estimate a β term for our intercept) and the second column in with numbers running from 1 to 89 (we want to estimate a β term for a slope indicating how the log-odds of DSR change with increasing season date; log-odds are discussed in Chapter 6).

Go ahead and run this model (you might call it '`B0 + B1 x Date`'). You can see that there's not much support for this model and that the real estimates don't indicate any large changes in DSR over the season. Even though in this case we don't see much seasonal variation in DSR, you can probably envision how this model might be of interest in other studies. Also, you might also be interested in pursuing curvilinear versions of this model (e.g., a model with a quadratic term) where the middle of the nesting season is expected to have the highest (or perhaps the lowest) DSR. If you did want to evaluate a quadratic term you would create a third column in the design matrix and fill it with the squared values of $1, 2, 3, \ldots, 89$, which, of course, are $1, 4, 9, \ldots, 7921$. (An easy way to do this is to build an Excel spreadsheet that has columns containing the values and to copy and paste them into the design matrix in **MARK**).

Model 6: DSR varies with nest age

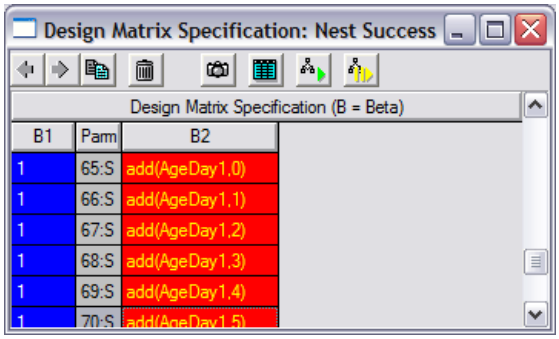
One might predict that nests will be more or less vulnerable as they age. This could be of interest because the species is known to provide predators with fewer and fewer cues each day as the nest ages. Perhaps this occurs because the species is known to increase nest attentiveness and to take fewer breaks from incubation as hatching nears. In contrast, in other species, one might predict that DSR is lower later in nesting when nestlings and feeding of nestlings may provide more cues to predators. Obviously, the predictions will vary by species. Regardless, we can use **MARK** to model a variety of models that allow DSR to vary with nest age.

Let's examine a simple model that allows DSR to follow a trend in accordance with nest age. To build this model, we first need to let DSR be different on each day of the nesting season. To do this, we once again need a PIM that is numbered from 1 to 89. Next, open a Design Matrix with 2 columns. Fill the first column of the design matrix with 1's (our intercept). Now for something that is new, at least in this chapter. We want to build a model that contains each nest's age on each day of the nesting season. Fortunately, we have the covariate `AgeDay1`, which you may recall tells us the age of the nest on the 1st day of the nesting season (a negative value for most of the nests!). We will now use this covariate and one of **MARK**'s handy design matrix functions to fill in our design matrix.

In the **MARK** Help file (or in Section 11.4 of Chapter 11 of this book), you should read over the section on *design matrix functions* - you can find it by searching for '`Design Matrix Add and Product Functions`' in the Help File's Index. We will use the '`add`' function, which not surprisingly is a function that adds 2 arguments together. To create an individual covariate that is each nest's age on each day of

the nesting season, we will start with AgeDay1 and simply add 0 to it on the first day of the nesting season, 1 to it on the 2nd day of the nesting season, 2 to it on the 3rd day of the nesting season,. . ., and 88 to it on the 89th day of the nesting season. The actual function takes the form 'add(argument1,argument2)' where the 2 arguments in this case are AgeDay1 and a continuous string of discrete numbers from 0 to 88. Some of you may be concerned about the fact that negative values were entered for most nest ages on day 1 of the nesting season. But, the negative values are never actually used to compute a survival, because a nest doesn't enter the likelihood until the 'add(AgeDay1,x)' value is zero, assuming that the correct age on day 1 (i.e., the correct negative value) has been entered. This business of adding to a negative age may be conceptually difficult at first but it does work!

The completed design matrix appears as follows:



Go ahead and run this model (you might call it 'B0 + B1 x Nest Age'). Be sure to uncheck the 'Standardize Individual Covariates' box, because otherwise the adding of the values in each row of the design matrix will not perform as you are expecting. You'll see that the results indicate that DSR might vary with nest age. But first, we might just want to run a few models that consider multiple covariates at once.

Note that you can also construct a model with age in a quadratic (or higher power function). Simply add a third column to the design matrix, and enter 'power(add(AgeDay1,0),2)' in the 1st row of the 3rd column, 'power(add(AgeDay1,1),2)' in the 2nd row of the 3rd column, and so on, ending with 'power(add(AgeDay1,88),2)' in the 89th row of the 3rd column. (As noted above, in both Chapter 11 of this book and in **MARK**'s help file for design matrix functions, there are instructions for how to create such a column efficiently in Excel using Excel's 'concatenate' function).

Model 7: models with multiple covariates

Let's imagine that the researchers had predicted *a priori* that DSR might vary with nest age and each of the various habitat measures (Habitat Type, Robel, and PpnGrass). So, let's build 3 more models by simply adding to the design matrix used for the 'B0 + B1 x Nest Age' model. We can do this using the 'Add Column' function (or the combination of the ctrl key and the 'A' key) while the design matrix window is active. Let's go ahead and run these 3 models and call them: (1) 'B0 + B1 x Nest Age + B2 x Robel', (2) 'B0 + B1 x Nest Age + B2 x PpnGrass', and (3) 'B0 + B1 x Nest Age + B2 x Native + B3 x Planted + B4 x Wetland'.

Results Browser: Nest Success

Model	AICc	Delta AICc	AICc Weight	Model Likelihood	No. Par.	Deviance
{B0 + B1 x Nest Age + B2 x PpnGrass}	1563.0095	0.0000	0.46477	1.0000	3	1557.0056
{B0 + B1 x Nest Age}	1564.0660	1.0565	0.27405	0.5896	2	1560.0640
{B0 + B1 x Nest Age + B2 x Robel}	1565.9055	2.8960	0.10924	0.2350	3	1559.9016
{B0 + B1 x Nest Age + B2 x Native + B3 x Planted + B4 x Wetland}	1567.3435	4.3340	0.05323	0.1145	5	1557.3337
{B0 + B1 x PpnGrass}	1567.3678	4.3583	0.05258	0.1131	2	1563.3658
{B0}	1569.1166	6.1071	0.02193	0.0472	1	1567.1159
{B0 + B1 x Robel}	1570.7749	7.7654	0.00957	0.0206	2	1566.7729
{B0 + B1 x Date}	1570.8265	7.8170	0.00933	0.0201	2	1566.8245
{B0 + B1 x Native + B2 x Planted + B3 x Wetland}	1571.9571	8.9476	0.00530	0.0114	4	1563.9506

It seems pretty clear that it's worth considering nest age and that `PpnGrass` appears to be important as well. We'll look at the details of model results a bit more below, but first, let's consider another factor possibly affecting DSR that one might want to investigate.

17.4.1. Models that consider observer effects on DSR

Although we won't go into all the details here, it seems worth pointing out that **MARK** also allows one to readily evaluate possible effects of observer visits to nests on DSR if the appropriate information is entered in the encounter history. In the case of the mallard study, 89 individual covariates could be added to the encounter history that simply are coded as 0 or 1 and that indicate whether a nest was visited on each day of the nesting season or not. These 89 covariates might be called something like `VisitDay1`, `VisitDay2`,..., `VisitDay89`. The idea being that we could then build a design matrix that modeled DSR as a function of whether or not a nest was visited on a given day. For example, if these covariates were included in the dataset, you could build a design matrix like the following:

Design Matrix Specification: Nest Success

Design Matrix Specification (B = Beta)

B1	Parm	B2
1	65:S	VisitDay1
1	66:S	VisitDay2
1	67:S	VisitDay3
1	68:S	VisitDay4
1	69:S	VisitDay5
1	70:S	VisitDay6

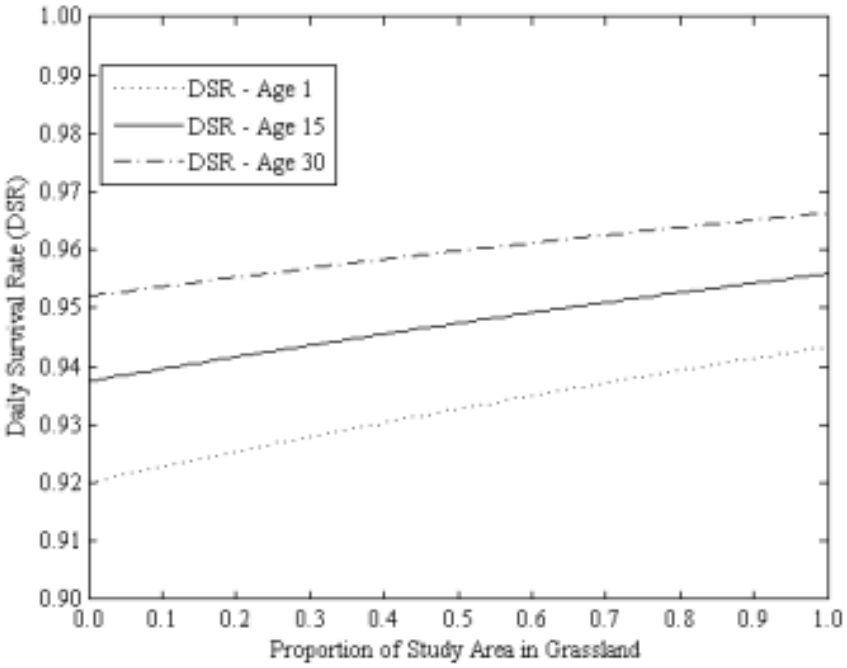
Further, you could include consideration of nest visits at the same time that other covariates were also considered. In cases where there is concern that nest visits may affect DSR on the day of the visit, such models may be of considerable interest. Other versions of such models could be built to consider more complex effects of visits on DSR, e.g., effects that are strongest immediately after a visit but that persist for multiple days.

Also recognize that you definitely do not want to use the 'Standardize Individual Covariates' option with the model shown in the above design matrix. If you were to standardize the individual covariates, each of the 89 variables `VisitDay1` through `VisitDay89` would be standardized differently, and as a result, the model would be nonsense. That is, the β_i value (β_2) would be multiplying variables with

different meanings in each row. So, by not standardizing the individual covariates, the meaning of `VisitDay1` is the same as the meaning of `VisitDay2`, etc.

17.5. Model results

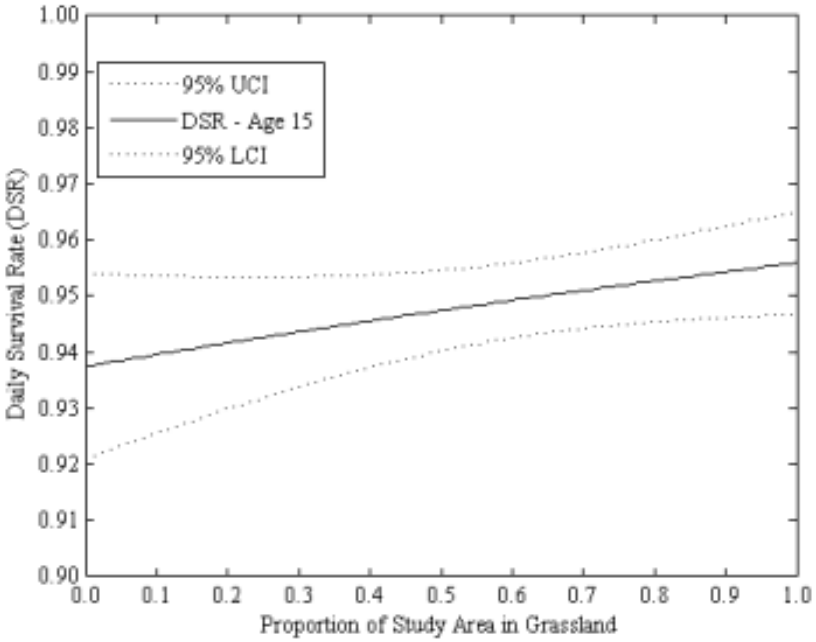
The most parsimonious model of DSR included `Nest Age` and `PpnGr`. This model was 1.06 AIC_c units better than the second-best model, which included `Nest Age` but not `PpnGr`, and was > 2.90 AIC_c units better than all other models evaluated. Models that held daily survival probability constant or simply allowed it to vary by habitat type, i.e., the only model types that have been used in many recent publications on nest survival, received little support ($\Delta AIC_c > 6.11$). The best model indicated that DSR increased with nest age ($\hat{\beta} = 0.0188, SE = 0.008$) and grassland extent ($\hat{\beta} = 0.369, SE = 0.211$), as can be seen in the following graphic (*note*: individual covariate values were not standardized in the analysis that produced these estimates so that actual nest ages and levels of `PpnGrass` could be used when producing the estimates; convergence was not problematic here).



When the logit link is used, our estimate of DSR for any combination of nest age and `PpnGrass` is simply

$$\frac{\exp\left(\beta_0 + \beta_1(\texttt{Nest age}) + \beta_2(\texttt{PpnGrass})\right)}{1 + \exp\left(\beta_0 + \beta_1(\texttt{Nest age}) + \beta_2(\texttt{PpnGrass})\right)}$$

So, for example, the estimated DSR for age=15:



So, the estimates presented in the figure can readily be obtained by substituting in the estimates of the β's and various values of `nest age` and `PpnGrass`.

Of course, one would want to see some measure of uncertainty on these graphs as well. So, it is worth considering how we can obtain estimates of the SE associated with various estimates of DSR. This is a bit more complicated because now there are 3 estimated β's and an associated 3×3 variance-covariance matrix for those estimates. One method of incorporating the variances and covariances associated with the 3 β's is the 'Delta method' (Seber 1982). Although we won't go into the use of the Delta method in this chapter (it is described in detail in Appendix 2), the figure (above) provides estimates of DSR and 95% confidence intervals estimated by the delta method for 15-days-old nests for different levels of `PpnGrass`.

17.6. Individual covariates and design matrix functions

In chapter 11 (section 11.4) we introduced several special functions that you can use in the design matrix, to make it easier to build specific models. These functions are especially useful for building complex models involving individual covariates. In Chapter 11 (section 11.4), there is a fairly comprehensive review of how to use several of these (specifically, the `add` and `ge` function) for a nest survival analysis which has fairly common design considerations.

17.7. Additional applications of the nest success model

Another application of the nest success model is to estimate the survival of radio-tracked animals when the animals are not monitored in discrete intervals, as required by the known-fate data type (see the

preceding chapter on *known-fate analysis*). Consider that such data are no different than a set of nests where all the nests are not visited on the same day. A DSR is estimated for each day of the study based on the sample of animals available on that day, and the exact day of death is not required (just as the exact day that a nest was destroyed is not known). We call these kinds of data 'Ragged Telemetry Data' because the sampling scheme is ragged, but useful estimates can still be obtained.

17.8. Goodness of fit and nest survival

Like the known fate data type, the saturated model for the nest survival data type is a useful model. Because the saturated model fits the data perfectly, there is no GOF test provided in Program **MARK**. However, a goodness of fit test for nest survival data was developed recently by Sturdivant *et al*. (2007), and it can be implemented with published **SAS** code, but not in **MARK**. Additional details, including code and sample datasets, can be found at: `http://www.montana.edu/rotella/nestsurv/`.

17.9. Summary

For nest survival data, Program **MARK** provides an excellent alternative to traditional constant-survival methods that have been used in most studies of nest survival to date. The methods shown in this chapter (1) can be used to conduct analyses of stratified data (appropriate if the simplifying assumptions of constant survival apply) and provide estimates that are almost identical to Mayfield estimates (or various refinements), (2) permit comparisons of survival probabilities among groups, (3) allow a much broader variety of covariates and competing models to be evaluated, and (4) should be employed in most nest-survival studies. It is worth noting that these methods can also be used for analyzing survival data collected from radiomarked individuals using ragged (uneven) intervals among animals and over time. The **MARK** help file provides more comments on this topic.

Despite these advances, further analysis improvements would be useful. Improved methods of estimating goodness-of-fit and for detecting and estimating overdispersion, or extra-binomial variation, would be useful given that a variety of factors may cause overdispersion. Nest-success data are commonly collected according to multilevel designs that result in grouped data, e.g., multiple observations on at least some nests, multiple nests per site, and multiple sites within each year. Thus, undetermined random effects of individuals, sites, and years could cause overdispersion or within-group correlations in daily survival probabilities, e.g., nest fates from multiple nests from within a colony or from a given study plot may not be independent. In addition, the spatial clustering of covariate levels could generate spatial correlation in nest survival rates and thus cause overdispersion. The random-effects model described by Rotella *et al*. (2004) can estimate random effects due to one source, e.g., site. However, even these methods do not accommodate multi-level nonlinear mixed models (e.g., some random effects associated with site, some associated with year, and others associated with individual nests), although, as mentioned above, they will be of interest in some studies (*note*: This is a good example of a situation where the new MCMC tool in **MARK** could be used, but further discussion of that complex topic is beyond the scope of this chapter). Finally, in some studies, uncertainty will exist about nest ages and when transitions among nest stages occur (Williams *et al*. 2002). This problem has been addressed for stratified data (Stanley 2000, 2004) but not yet for data with more complex covariates.

References

Bart, J., and D. S. Robson. 1982. Estimating survivorship when the subjects are visited periodically. *Ecology*, **63**, 1078-1090.

Dinsmore, S. J., G. C. White, and F. L. Knopf. 2002. Advanced techniques for modeling avian nest survival. *Ecology*, **83**, 3476-3488.

Klett, A. T., H. F. Duebbert, C. A. Faanes, and K. F. Higgins. 1986. Techniques for studying nest success of ducks in upland habitats in the prairie pothole region. United States Fish and Wildlife Service Resource Publication No. 158.

Hensler, G. L., and J. D. Nichols. 1981. The Mayfield method of estimating nest success: A model, estimators and simulation results. *Wilson Bulletin*, **93**, 42-53.

Johnson, D. H. 1979. Estimating nest success: the Mayfield method and an alternative. *Auk*, **96**, 651-661.

Mayfield, H. F. 1961. Nesting success calculated from exposure. *Wilson Bulletin*, **73**, 255-261.

Mayfield, H. F. 1975. Suggestions for calculating nest success. *Wilson Bulletin*, **87**, 456-466.

Rotella, J. J., S. J. Dinsmore, and T. L. Shaffer. 2004. Modeling nest-survival data: a comparison of recently developed methods that can be implemented in **MARK** and **SAS**. *Animal Biodiversity and Conservation*, **27**, 187-204.

Seber, G. A. F. 1982. *The Estimation of Animal Abundance and Related Parameters. 2nd ed*. Macmillan, New York, New York, USA. 654 pp.

Stanley, T. R. 2000. Modeling and estimation of stage-specific daily survival probabilities of nests. *Ecology*, **81**, 2048-2053.

Stanley, T. R. 2004. Estimating stage-specific daily survival probabilities of nests when nest age is unknown. *Auk*, **121**, 134-147.

Stephens, S. E. 2003. The influence of landscape characteristics on duck nesting success in the Missouri Coteau Region of North Dakota. Ph.D. Dissertation. Montana State University.

Sturdivant, R. X., J. J. Rotella, and R. E. Russell. 2007. A smoothed residual based goodness-of-fit statistic for nest-survival models. *Studies in Avian Biology*, **34**, 45-54.

Williams, B. K., J. D. Nichols, and M. J. Conroy. 2002. *Analysis and management of animal populations: modeling, estimation, and decision making*. Academic Press, New York.

CHAPTER 18

Mark-resight models

Brett McClintock, *NOAA National Marine Mammal Laboratory, Seattle, Washington, USA*

Mark-resight methods constitute a slightly different type of data than found in traditional mark-recapture, but they are in the same spirit of accounting for imperfect detection towards reliably estimating demographic parameters (see White & Shenk 2001 for a thorough explanation of how these data are collected, and McClintock *et al.* 2009b, McClintock & White 2009, and McClintock & White 2012 for full details of the models). Like the other mark-recapture models in **MARK**, this approach models encounters (resightings) of marked individuals, but they also incorporate additional data via sightings of unmarked individuals into the estimation framework. Mark-resight data may be used to estimate abundance (N) in a fashion analogous to the closed capture models of Otis *et al.* (1978). When sampling is under the robust design, mark-resight data may be used to estimate abundance, apparent survival, and transition probabilities between observable and unobservable states in a fashion analogous to the closed capture robust design models of Kendall, Pollock & Brownie (1995) and Kendall, Nichols & Hines (1997).

These models assume some individuals have been marked prior to sampling, and sampling occasions consist of sighting surveys (instead of capture periods). The main advantage of this approach is that because costs associated with marking and recapturing can be minimized, it can in many circumstances be a less invasive and less expensive alternative to traditional mark-recapture as a means for monitoring. With limited funds and resources, mark-resight can be appealing to researchers because costs associated with capture are generally the most expensive aspects of mark-recapture studies. Not only can the financial burden of mark-recapture be discouraging for long-term population monitoring, but capture is also the most hazardous aspect for the animals and may unduly influence the attributes of scientific interest. If field-readable marks are feasible, mark-resight can substantially reduce stress to species because they can be observed at a distance with minimal disturbance after the initial marking period. This can be of particular concern when working with threatened, endangered, or sensitive species.

The methods require that the number of marked individuals in the population during sampling be known exactly or can at least be reliably estimated. If sampling during sighting occasions is without replacement (i.e., any single individual may only be sighted once per distinct occasion) and the number of marked individuals in the population available for resighting is known exactly, then the mixed logit-normal mark-resight model (McClintock *et al.* 2009b) may be employed to estimate N. If the mixed logit-normal model is appropriate but the population of interest within the study area is known to lack geographic closure (e.g., from telemetry data for the marked population), the immigration-emigration logit-normal model may be used to estimate N (or density). Alternatively, if sampling within sighting occasions is with replacement or the exact number of marked individuals in the population is unknown,

© Cooch & White (2014)

the Poisson-log normal mark resight model (McClintock *et al.* 2009a) may be used to estimate N. If permanent field-readable marks are used but the number of marks is not known, then mark-resight data collected under the closed robust design may be analyzed with the Poisson-log normal model in a fashion analogous to the regular mark-recapture robust design for estimating apparent survival (φ), transition probabilities between observable and unobservable states (γ'' and γ'), and N (McClintock & White 2009).

These models were developed as reliable and more efficient alternatives to the mark-resight models previously available in Program **NOREMARK** (White 1996). Similar to the mark-recapture models in **MARK**, they provide a framework for information-theoretic model selection and multimodel inference based on AIC (Burnham & Anderson 2002) and the utilization of individual or environmental covariates on parameters. However, because the nature of mark-resight data is somewhat different than that of mark-recapture, a different format for the encounter history files has been developed to address this. Explanations of the various models and their **MARK** encounter history file formats are detailed below. The encounter history and results files referenced here accompany **MARK**. Following the explanations of the models and their **MARK** encounter history files, some general suggestions are provided for performing an analysis with these models in **MARK**. But first, a little more background on mark-resight.

18.1. What is mark-resight?

The basic premise behind mark-resight is fairly simple. First, some field-readable marks are introduced into the population. Then encounter data are collected (via non-invasive sighting surveys) on both the marked ***and*** unmarked individuals in the population. Lastly, the data are analyzed to estimate abundance (N) and/or related demographic parameters (φ, γ', γ''). Pretty simple, right? As usual, the complications lie in the particulars.

Initially, the focus of mark-resight was on utilizing radio-marked individuals to estimate closed population abundance. This dependency on radio-collars arose because of a need to know the exact number of marked individuals in the population. One of the simplest mark-resight models of abundance is the classic Lincoln-Petersen estimator:

$$\hat{N} = \frac{m_1 n_2}{m_2},$$

where m_1 is the number of marked animals in the population, n_2 is the total number of marked and unmarked animals seen, and m_2 is the number of marked animals seen. Users of Program **NOREMARK** are probably familiar with other mark-resight models of abundance, such as the joint hypergeometric estimator (Bartmann *et al.* 1987), the Minta-Mangel estimator (Minta & Mangel 1989), the immigration-emigration joint hypergeometric estimator (Neal *et al*. 1993), and Bowden's estimator (Bowden & Kufeld 1995). Arnason, Schwarz & Gerrard (1991) developed a mark-resight model of abundance when the number of marked individuals in the population is unknown. These contributions were the motivation for developing a more general suite of mark-resight estimators that would fit into the flexible modeling framework that **MARK** provides.

There are several things to consider when deciding to use the mark-resight models in **MARK**. As with all mark-recapture studies, a population of interest must first be defined (both in space and time). For starters, we will assume this population is geographically and demographically closed, and abundance for a single period of time is the only item of interest. The simplest issue relevant to mark-resight is whether or not individuals in the population can possess field-readable marks. You're unlikely to use mark-resight on *Peromyscus*, but it has been applied to many different species including ursids, canids, badgers, ungulates, prairie dogs, snail kites, owls, robins, and grouse. Field-readable marks may come in

many forms, including collars, bands, paint, dye, or natural patterns. The marks may be temporary (e.g., paint or dye on fur) or permanent, but no (unknown) marks may be lost during the sampling period of interest. An important distinction for the mark-resight models in **MARK** is whether the marks are individually identifiable or not. Much more information (and flexibility) can be attained through the use of individually identifiable marks, particularly if individual sighting probability heterogeneity is of concern. However, this methodology may still be employed if individually identifiable marks are not feasible (e.g., due to species or monetary constraints).

If field-readable marks are possible, then marked individuals must be introduced into the population before any sighting data can be collected. This is typically done via a capture event (but not necessarily). Whatever the marks and however they are introduced, **the most fundamental assumption of mark-resight is that the subset of the population selected for marking is representative of the entire population in terms of sighting probabilities**. A strategy typically employed to satisfy this condition is the use of a different method to randomly select marked individuals than is used for the sighting surveys. This may seem obvious, but mark-resight has often been applied (inappropriately) when the marked population was selected based on sightability. One must also make sure that the marks themselves do not affect sightability. If the tags on the marked individuals are so eye-catching that they make the marked population much more sightable than the unmarked individuals, then the sighting probabilities will be overestimated (and population size therefore underestimated). For example, suppose fish were marked with bright fluorescent tags on the dorsal side (Fig. 18.1) and sighted from a bridge above. If the bright tags made the marked individuals sightable from depths where unmarked individuals were not, then there would be a problem.

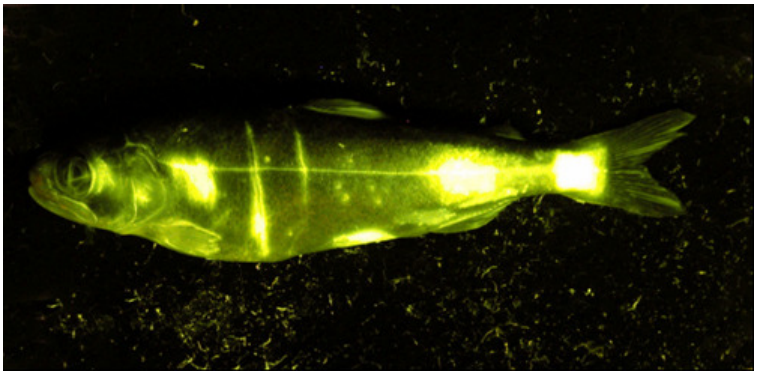


Figure 18.1: *Example of a 'tagged' organism which may be significantly more 'visible' than untagged individuals, which would potentially bias estimates of population size.*

Once marks have been introduced into the population, an important piece of information becomes how many marked individuals are alive and in the study area. If the number of marked individuals available for resighting is known exactly, this can be very useful information for estimation (particularly when individual sighting heterogeneity is a serious issue). The number of marks in the population is commonly determined via radio or GPS collars that emit a mortality signal. Another way this is accomplished is by conducting the marking period immediately prior to the collection of sighting data, such that it can be reasonably assumed that no marked individuals died or emigrated between the capture event and the sighting surveys. When marked individual mortality or movement cannot be monitored and sufficient time has passed since the original introduction of marks, then the exact number of marks will usually be unknown.

The actual sighting data are collected during visual surveys within the study area. All sightings of

marked and unmarked individuals in the population are recorded. If individually identifiable marks are used, then the individual identities of marked individuals are also recorded. **The sighting surveys themselves come in two basic flavors: sampling with or without replacement**. If sampling is without replacement, then each individual in the population can be seen at most once within each of the distinct sampling occasions (as in mark-recapture). However, in many circumstances sampling must be with replacement. This arises when sampling cannot be divided into distinct occasions where individuals can only be sighted once, such as when studying a highly mobile species or using camera traps. Sampling with replacement differs from other mark-recapture sampling because here sighting occasions need not be distinct, and consideration is given only to some closed period of sampling. When sampling with replacement, particular care must be taken to ensure that all sightings of a marked or unmarked individual are recorded, irrespective of marked status or any previous sightings. For example, suppose one were to stop counting resightings of a marked individual after it had already been resighted during the sampling period, but continued to record all unmarked sightings (because one cannot know whether or not an unmarked individual had been previously sighted during the sampling period). This would violate the must fundamental assumption of mark-resight because the marked individual sighting probabilities would not be representative of the unmarked individual sighting probabilities.

Sighting probabilities are modeled with mark-resight estimators just as capture probabilities are modeled with mark-recapture estimators. This means group, temporal, or individual covariates may be utilized to describe the detection process. Individual sighting heterogeneity is also an important issue because failure to account for it may result in underestimates of abundance (if the number of marks is unknown) and overestimates of precision. Individual heterogeneity may only be accounted for if marks are individually identifiable.

As is the case in most monitoring programs, let's now consider more than a single closed period of interest. We will adopt the terminology of the robust design (Kendall, Pollock & Brownie 1995; Kendall, Nichols & Hines 1997), where data are collected across both closed and open sampling periods. The open periods refer to the encounter process between 'primary' sampling intervals, where each primary interval consists of 'secondary' sighting occasions. The time periods between the secondary sighting occasions within a primary interval must be of short enough duration for the assumption of closure to be satisfied (although this may in some circumstances be relaxed – see the next paragraph). As noted before, if sampling is with replacement, then we are not concerned with distinct secondary sighting occasions, but rather some closed period of secondary sampling during each of the primary intervals. New marks may be introduced to the population at any time during the open periods, but no marks may be added during the closed periods (except when using the immigration-emigration logit-normal model).

The issue of closure deserves a bit of attention before getting into the specifics on implementing the logit-normal, immigration-emigration logit-normal, and Poisson-log normal mark-resight models in **MARK**. When the population of interest is both geographically and demographically closed, then the estimates of abundance produced by all of the mark-resight models are exactly what we think they are: the population size residing within the study area during the period of interest. If the population is not geographically closed (i.e., individuals move in and out of the study area), then there are two notions of 'population' for the study area. There is the population that actually resides within the study area during the period of interest (N), but there is also a 'super population' of individuals associated with the study area during the period of interest (N^*). This distinction is important, because the latter is unsuitable for addressing questions related to population density. When geographic closure is violated, then the mixed logit-normal and Poisson-log normal mark-resight models produce estimates of N^*. For this reason, the immigration-emigration logit-normal model was developed as a means for estimating both N and N^* when geographic closure is violated. When demographic closure is violated

(i.e., individuals may die or permanently emigrate independent of mark status), all of the models will produce estimates of the population size at the beginning of the sampling period of interest. Because the lack of geographic or demographic closure may induce non-negligible levels of individual sighting heterogeneity, we suggest that heterogeneity models be explored when these violations are suspected (this requires individually identifiable marks).

18.2. The mixed logit-normal mark-resight model

To be used when sampling is without replacement within secondary sampling occasions and the number of marked individuals in the population available for resighting is known exactly. Marks may or may not be individually identifiable. See McClintock *et al.* (2009b) for full details.

Data:

t = the number of primary sampling intervals

k_j = the number of secondary sampling occasions (without replacement) during the primary interval j

n_j = the exact number of marked individuals in the population during primary interval j

m_{ij} = $\sum_{s=1}^{n_j} \delta_{sij}$ = total number of marked individual sightings during secondary occasion i of primary interval j

T_{uj} = total number of unmarked individual sightings during primary interval j

δ_{sij} = Bernoulli random variable indicating sighting ($\delta_{sij} = 1$) or no sighting ($\delta_{sij} = 0$) of marked individual s on secondary occasion i of primary interval j (this only applies when individually identifiable marks are used)

ϵ_{ij} = total number of marks seen that were not identified to individual during secondary occasion i of primary interval j (this only applies when individually identifiable marks are used)

Parameters:

N_j = population size or abundance during primary interval j

p_{ij} = intercept (on logit scale) for mean resighting probability of secondary occasion i during primary interval j. If there is no individual heterogeneity ($\sigma_j = 0$), once back-transformed from the logit scale the real parameter estimate can be interpreted as the mean resighting probability

σ_j^2 = individual heterogeneity level (on the logit scale) during primary interval j (i.e., the variance of a random individual heterogeneity effect with mean zero)

Derived Parameter:

μ_{ij} = overall mean resighting probability for secondary occasion i of primary occasion j. This parameter is derived as a function of p_{ij}, σ_j^2, and ϵ_{ij}. Note that when $\sigma_j = 0$ and $\epsilon_{ij} = 0$, then the real parameter estimate for p_{ij} is identical to the derived parameter estimate for μ_{ij}.

18.2.1. No individually identifiable marks

If a known number of marks are in the population, but the marks are not individually identifiable, then the data for the mixed logit-normal model are t, k_j, n_j, m_{ij}, and T_{uj}. These are the same data as for the

joint hypergeometric estimator (JHE) previously available in Program **NOREMARK** (White 1996), but the mixed logit-normal model can be a more efficient alternative because it can borrow information about resighting probabilities across primary intervals or groups (McClintock *et al.* 2009b). Note that because no information is known about individual identities, individual heterogeneity models cannot be evaluated with these data (i.e., $\sigma_j = 0$) and the probability of any individual being resighted on secondary occasion i of primary interval j is p_{ij}.

Suppose there is only one group and $t = 3, k_j = 4, n_1 = 30, n_2 = 33, n_3 = 32, m_{11} = 8, m_{21} = 9, m_{31} = 10, m_{41} = 5, m_{12} = 11, m_{22} = 10, m_{32} = 18, m_{42} = 9, m_{13} = 5, m_{23} = 10, m_{33} = 13, m_{43} = 8, T_{u1} = 96, T_{u2} = 68,$ and $T_{u3} = 59$.

Although no individual identities are known, these data may be summarized into artificial individual encounter histories similar to those of the mark-recapture robust design. The total number of unmarked individuals seen $\left(T_{uj}\right)$ must be entered after the encounter histories under the heading '`Unmarked Seen Group=1`' such that the resulting encounter history file would be:

```
/* No Individual Marks 1 group */
/* 12 occasions, 3 primary, 4 secondary each */

/* Begin Input File */

111111111111 5;
111011110111 3;
011011110110 1;
001011100110 1;
000010100010 1;
000000100010 2;
000000100000 5;
000000000000 12;
....00000000 2;
....0000.... 1;

Unmarked Seen Group=1;
96 68 59;

/* End Input File */
```

Notice the sums of the encounter history columns (when multiplied by the corresponding frequency) equal m_{ij} and the sums of the frequencies with non-missing entries (i.e., not '....') for each primary interval equals n_j. If this single group data were split into two groups, such that $n_1 = 17, n_2 = 19, n_3 = 18, m_{11} = 6, m_{21} = 6, m_{31} = 7, m_{41} = 4, m_{12} = 5, m_{22} = 5, m_{32} = 11, m_{42} = 5, m_{13} = 3, m_{23} = 7, m_{33} = 7, m_{43} = 7, T_{u1} = 48, T_{u2} = 40,$ and $T_{u3} = 20$ for the first group, and $n_1 = 13, n_2 = 14, n_3 = 14, m_{11} = 2, m_{21} = 3, m_{31} = 3, m_{41} = 1, m_{12} = 6, m_{22} = 5, m_{32} = 7, m_{42} = 4, m_{13} = 2, m_{23} = 3, m_{33} = 6, m_{43} = 1, T_{u1} = 48, T_{u2} = 28,$ and $T_{u3} = 39$ for the second group, a possible encounter history file would be:

```
/* No Individual Marks 2 groups */
/* 12 occasions, 3 primary, 4 secondary each */

/* Begin Input File */

111111111111 3 0;
111111110111 1 0;
```

```
111011110111 1 0;
111000100111 1 0;
001000100111 1 0;
000000100000 4 0;
000000000000 6 0;
....00000000 1 0;
....0000.... 1 0;
111111111111 0 1;
111011111110 0 1;
011011110110 0 1;
000011110010 0 1;
000011100010 0 1;
000010100010 0 1;
000000100000 0 1;
000000000000 0 6;
....00000000 0 1;

Unmarked Seen Group=1;
48 40 20;

Unmarked Seen Group=2;
48 28 39;

/* End Input File */
```

Notice here that the single group data has simply been split up into two group data. The encounter histories are followed by group frequencies just as in other **MARK** encounter history files for mark-recapture data. The twist is that the unmarked data must be entered separately for each group. Again, the sums of the encounter history columns (when multiplied by the corresponding group frequencies) equals m_{ij} for each group, and the sums of the frequencies with non-missing entries (i.e., not '....') for each primary interval equals n_j for each group.

The analysis using these encounter history data (`Logit_NoIndividualMarks_OneGroup.inp`) yielded the following results for the time-constant ($p_{ij} = p, \sigma_j = 0$) model in **MARK**:

```
                Real Function Parameters of {p(.) sigma(.)=0 N(t)}

                                                          95% Confidence Interval
Parameter                 Estimate       Standard Error      Lower           Upper
------------------------  -------------  --------------  --------------  --------------
   1:p Session 1          0.3064700       0.0236970       0.2620778       0.3547665
   2:sigma Session 1      0.0000000       0.0000000       0.0000000       0.0000000       Fixed
   3:N Session 1          108.02874       8.9593461       92.350040       127.65004
   4:N Session 2          88.188211       7.0136070       76.062792       103.72785
   5:N Session 3          79.846656       6.3659903       68.905724       94.031094
```

Note that σ_j must be fixed to zero for these data because heterogeneity models do not apply when marks are not individually identifiable. This is because no information is known about individual resighting probabilities, and the above encounter histories are artificial in that they don't actually refer to a real individual's encounter history (these artificial encounter histories are just a convenient and

consistent way to enter the data into **MARK**). Because there is no individual heterogeneity in the model, the real parameter estimate of p may be interpreted as the overall mean resighting probability (0.31 in this case).

18.2.2. Individually identifiable marks

If marks are individually identifiable, encounter histories are constructed just as for robust design mark-recapture data with the tk_j possible encounters representing δ_{sij} for individual s during secondary occasion i of primary interval j. However, now it is possible to have an individual identified as marked, but not to individual identity. A marked individual may be encountered but not be identified to individual when the mark was seen but the unique pattern or characters that identify the individual were obscured or too far away to read. These are the same data as could be used for Bowden's estimator (Bowden & Kufeld 1995) in Program **NOREMARK** (White 1996), but the logit-normal model can be more efficient because information about resighting probabilities may be borrowed across primary intervals, and it does not require investment in individual heterogeneity parameters unless deemed necessary by the data (McClintock *et al.* 2009b). If an individual was not known to be in the population during any primary interval j, then missing values (.) are included for all k_j secondary occasions of that interval in the encounter history. The total number of marks seen but not identified to individual during secondary occasion i of primary interval j (ϵ_{ij}) are entered sequentially $\left(\epsilon_{11}, \epsilon_{21}, \ldots, \epsilon_{k_1 1}, \ldots, \epsilon_{1t}, \epsilon_{2t}, \ldots, \epsilon_{k_t t}\right)$ with each entry separated by a space. Using the data from the previous single group example but with ϵ = `(0,0,0,0,1,1,1,0,0,3,0,1)` entered after the unmarked data under the heading '**`Marked Unidentified Group=1;`**', one possible encounter history file would be:

```
/* Individual Marks 1 Group */
/* 12 occasions, 3 primary, 4 secondary each */

/* Begin Input File */

001001000011 1;
000000100110 1;
010000000110 1;
0000........ 1;
....01101101 1;
000010000000 1;
001100100000 1;
001011100011 1;
000010000010 1;
010001100000 1;
000000000010 1;
001010010110 1;
101000100000 1;
....01001110 1;
010000100000 1;
11001000.... 1;
000100000000 1;
100000101011 1;
000011010000 1;
000100000000 1;
111000100001 1;
010000111001 1;
101000110000 1;
```

```
100001100010 1;
....00010000 1;
101000010010 1;
0000........ 1;
010000101000 1;
000110100000 1;
011000000000 1;
010011110010 1;
000010110000 1;
101100000001 1;
....00010110 1;
....11100100 1;

Unmarked Seen Group=1;
96 68 59;

Marked Unidentified Group=1;
0 0 0 0 1 1 1 0 0 3 0 1;

/* End Input File */
```

Note that the sums of each column $\sum_{s=1}^{n_j} \delta_{sij} = m_{ij} - \epsilon_{ij}$. The last two encounter histories are for individuals that were not marked and known to be in the population until immediately prior to the second primary interval. The fourth encounter history from the top represents an individual who was marked and known to be in the population during the first primary interval (when it was resighted 0 times), but known to have not been marked and in the population during the second or third primary intervals. This could be because the individual was known to have died, emigrated, or lost its mark. Similar to other **MARK** encounter history files, the histories may pertain to multiple groups and include individual covariates. Splitting the above data into two groups, the above encounter history file could look like:

```
/* Individual Marks 2 Groups */
/* 12 occasions, 3 primary, 4 secondary each */

/* Begin Input File */

001001000011 0 1;
000000100110 1 0;
010000000110 1 0;
0000........ 1 0;
....01101101 1 0;
000010000000 0 1;
001100100000 1 0;
001011100011 0 1;
000010000010 0 1;
010001100000 0 1;
000000000010 0 1;
001010010110 1 0;
101000100000 1 0;
....01001110 1 0;
010000100000 1 0;
11001000.... 1 0;
```

```
000100000000 1 0;
100000101011 1 0;
000011010000 1 0;
000100000000 0 1;
111000100001 1 0;
010000111001 0 1;
101000110000 1 0;
100001100010 0 1;
....00010000 0 1;
101000010010 0 1;
0000........ 0 1;
010000101000 0 1;
000110100000 1 0;
011000000000 1 0;
010011110010 1 0;
000010110000 0 1;
101100000001 1 0;
....00010110 1 0;
....11100100 0 1;

Unmarked Seen Group=1;
48 40 20;

Unmarked Seen Group=2;
48 28 39;

Marked Unidentified Group=1;
0 0 0 0 0 1 1 0 0 1 0 1;

Marked Unidentified Group=2;
0 0 0 0 1 0 0 0 0 2 0 0;

/* End Input File */
```

Notice the encounter histories are followed by group frequencies the same way as they are in all other **MARK** encounter history files.

Because marks are individually identifiable, individual heterogeneity models may be explored with these data. Here, individual heterogeneity is modeled as a random effect with mean zero and unknown variance σ_j^2. These encounter history data (`Logit_IndividualMarks_OneGroup.inp`) yielded the following results for the time-constant individual heterogeneity ($p_{ij} = p, \sigma_j = \sigma$) model in **MARK**:

```
                Real Function Parameters of {p(.) sigma(.) N(t)}

                                                          95% Confidence Interval
Parameter                 Estimate       Standard Error      Lower           Upper
------------------------  -------------  --------------  --------------  --------------
   1:p Session 1          0.2786641       0.0273014       0.2284108       0.3351710
   2:sigma Session 1      0.4766088       0.2707817       0.1690244       1.3439241
   3:N Session 1          112.97626       10.415916       94.940988       136.02025
   4:N Session 2          87.429921       6.9734104       75.386318       102.89558
   5:N Session 3          77.935945       6.0515938       67.521842       91.403200
```

If one wanted to report an overall mean resighting probability for this model, then the derived parameter μ_{ij} may be obtained:

Estimates of Derived Parameters
Mean Resighting Rate Estimates of {p(.) sigma(.) N(t)}

Grp.	Occ.	Mu-hat	Standard Error	95% Confidence Interval Lower	Upper
1	1	0.2880297	0.0247720	0.2420014	0.3388985
1	2	0.2880297	0.0247720	0.2420014	0.3388985
1	3	0.2880297	0.0247720	0.2420014	0.3388985
1	4	0.2880297	0.0247720	0.2420014	0.3388985
1	5	0.3183328	0.0247720	0.2718623	0.3687242
1	6	0.3183328	0.0247720	0.2718623	0.3687242
1	7	0.3183328	0.0247720	0.2718623	0.3687242
1	8	0.2880297	0.0247720	0.2420014	0.3388985
1	9	0.2880297	0.0247720	0.2420014	0.3388985
1	10	0.3817797	0.0247720	0.3345418	0.4313640
1	11	0.2880297	0.0247720	0.2420014	0.3388985
1	12	0.3192797	0.0247720	0.2727964	0.3696574

Even though the model included a constant p and σ for all occasions, there is some slight variation in μ_{ij} due to marked individuals not being identified to individual identity (ϵ_{ij}) on several occasions.

The time-constant model with no heterogeneity ($p_{ij} = p, \sigma_j = 0$) yields:

Real Function Parameters of {p(.) sigma(.)=0 N(t)}

Parameter	Estimate	Standard Error	95% Confidence Interval Lower	Upper	
1:p Session 1	0.2881305	0.0232879	0.2447124	0.3358270	
2:sigma Session 1	0.0000000	0.0000000	0.0000000	0.0000000	Fixed
3:N Session 1	112.98732	9.7939840	95.902227	134.50170	
4:N Session 2	87.446686	6.5355052	76.068609	101.83068	
5:N Session 3	77.954031	5.6720754	68.112315	90.477916	

Estimates of Derived Parameters
Mean Resighting Rate Estimates of {p(.) sigma(.)=0 N(t)}

Grp.	Occ.	Mu-hat	Standard Error	95% Confidence Interval Lower	Upper
1	1	0.2881305	0.0232879	0.2447124	0.3358270
1	2	0.2881305	0.0232879	0.2447124	0.3358270
1	3	0.2881305	0.0232879	0.2447124	0.3358270
1	4	0.2881305	0.0232879	0.2447124	0.3358270
1	5	0.3184336	0.0232879	0.2746235	0.3657090
1	6	0.3184336	0.0232879	0.2746235	0.3657090

```
1     7   0.3184336       0.0232879       0.2746235       0.3657090
1     8   0.2881305       0.0232879       0.2447124       0.3358270
1     9   0.2881305       0.0232879       0.2447124       0.3358270
1    10   0.3818805       0.0232879       0.3373910       0.4284434
1    11   0.2881305       0.0232879       0.2447124       0.3358270
1    12   0.3193805       0.0232879       0.2755591       0.3666438
```

As before, when $\sigma_j = 0$, the real parameter estimate of p may be interpreted as the overall mean resighting probability ignoring unidentified marks (0.29 in this case), but μ_{ij} is an overall mean resighting probability that takes unidentified marks into account. Notice that these results are different than the results from the same model when there were no individually identifiable marks. This is because the two versions (individually identifiable marks or not) of the mixed-logit normal model are only comparable when all marks are correctly identified to individual and σ_j is fixed to zero. Further, if one finds very little support for individual heterogeneity models (based on AIC_c) and has relatively many unidentified marks, it may be better to analyze the data as if there were no individually identifiable marks to begin with.

18.3. The immigration-emigration mixed logit-normal model

For use when the population of interest may not be geographically closed (i.e., individuals move in and out of the study area between secondary occasions of the primary sampling intervals). Because the study area is not closed, there is a 'super population' of individuals that use the area, but the population of interest may be that which actually resides within the study area at any given time. This distinction is important when density estimation is of concern. This model requires additional information on whether or not each marked individual was available for resighting within the study area for each secondary sampling occasion (e.g., from radio or GPS collars). One way this is commonly determined using radio-collars is by conducting an aerial survey locating all marked individuals immediately prior to each secondary sampling occasion, although the use of GPS collars may alleviate the need for such surveys. Once the presence or absence of all marked individuals within the study area is determined, secondary resighting occasions are conducted only within the boundaries of the study area. As with the regular mixed logit-normal model, sampling must be without replacement within secondary sampling occasions and the number of marked individuals in the population available for resighting must be known exactly for every secondary sampling occasion. Marks may or may not be individually identifiable (but individually identifiable marks are needed to investigate individual heterogeneity). Unlike the regular mixed logit-normal or the Poisson-log normal models (where new marks may be introduced only during the open periods), new marks may be introduced at any time (other than *during* a secondary sampling occasion) when using the immigration-emigration mixed logit-normal model. See McClintock & White (2012) for full details.

Data:

t = the number of primary sampling intervals

k_j = the number of secondary sampling occasions (without replacement) during primary interval j

n_j = the exact number of marked individuals in the population during primary interval j

$m_{ij} = \sum_{s=1}^{n_j} \delta_{sij}$ = total number of marked individual sightings during secondary occasion i of primary interval j

T_{uij} = total number of unmarked individual sightings during secondary occasion i of primary interval j

δ_{sij} = Bernoulli random variable indicating sighting ($\delta_{sij} = 1$) or no sighting ($\delta_{sij} = 0$) of marked individual s on secondary occasion i of primary interval j (this only applies when individually identifiable marks are used)

T_{ij} = number of marked animals in the super population during secondary occasion i of primary interval j. A marked individual is considered to be in the super population if it were located within the study area at least once during primary interval j

M_{ij} = number of marked animals that are actually in the study area during secondary occasion i of primary interval j

ϵ_{ij} = total number of marks seen that were not identified to individual during secondary occasion i of primary interval j (this only applies when individually identifiable marks are used)

Parameters:

N_j^* = super-population size utilizing the study area at any time during primary interval j

$\bar{N}_j$ = mean population size within the study area during primary interval j. Because this quantity is generally of more interest (e.g., for density estimation) than the population size within the study area during secondary occasion i of primary interval j (N_{ij}), **MARK** uses the reparameterization $N_{ij} = \bar{N}_j + \alpha_{ij}$ where $\sum_{i=1}^{k_j} \alpha_{ij} = 0$

α_{ij} = the difference (relative to $\bar{N}_j$) in population size within the study area during secondary occasion i of primary interval j. Because of the imposed constraint $\sum_{i=1}^{k_j} \alpha_{ij} = 0$, only $k_j - 1$ of the α_{ij} must actually be estimated for primary interval j.

p_{ij} = intercept (on logit scale) for mean resighting probability of secondary occasion i during primary interval j. If there is no individual heterogeneity ($\sigma_j = 0$), once back-transformed from the logit scale the real parameter estimate can be interpreted as the mean resighting probability

σ_j^2 = individual heterogeneity level (on the logit scale) during primary interval j (i.e., the variance of a random individual heterogeneity effect with mean zero)

Derived Parameter:

μ_{ij} = overall mean resighting probability for secondary occasion i of primary interval j. This parameter is derived as a function of p_{ij}, σ_j^2, M_{ij}, and ϵ_{ij}. Note that when $\sigma_j = 0$ and $\epsilon_{ij} = 0$, then the real parameter estimate for p_{ij} is identical to the derived parameter estimate for μ_{ij}.

18.3.1. No individually identifiable marks

If a known number of marks are in the population, but the marks are not individually identifiable, then the data for the immigration-emigration mixed logit-normal model are $t, k_j, T_{ij}, M_{ij}, m_{ij}$, and T_{uij}. These are the same data as for the immigration-emigration joint hypergeometric estimator (IEJHE) previously available in Program **NOREMARK** (White 1996), but the immigration-emigration mixed logit-normal model can be a more efficient alternative because it can borrow information about resighting probabilities across primary intervals. Note that because no information is known about individual identities, individual heterogeneity models cannot be evaluated with these data (i.e., $\sigma_j = 0$) and the probability

of any individual being resighted on secondary occasion i of primary interval j is p_{ij}.

Here we'll use vector notation because we must keep track of data for each secondary occasion of each primary interval, where any $\boldsymbol{x} = \{x_{11}, x_{21}, \ldots, x_{k_1 1}, x_{12}, x_{22}, \ldots, x_{k_2 2}, \ldots, x_{1t}, x_{2t}, \ldots, x_{k_t t}\}$. Now, suppose there is only one group and

$$
\begin{aligned}
t &= 3 \\
k_j &= 4 \\
\boldsymbol{n} &= \{27, 22, 18, 29, 28, 23, 20, 32, 31, 19, 21, 33\} \\
\boldsymbol{T} &= \{28, 29, 30, 30, 30, 33, 33, 33, 33, 34, 34, 34\} \\
\boldsymbol{m} &= \{17, 15, 9, 8, 16, 14, 9, 13, 11, 14, 13, 16 \\
\boldsymbol{T_u} &= \{264, 161, 152, 217, 217, 160, 195, 159, 166, 152, 175, 190\}
\end{aligned}
$$

These data show that marks were introduced into the population between secondary sampling occasions at some point for all three primary intervals. For example, one mark was introduced between the first ($T_{11} = 28$) and second ($T_{21} = 29$) secondary occasions of the first primary interval. Of these marked individuals in the super population using the study area, $n_{11} = 27$ and $n_{21} = 22$ marked individuals, respectively, were actually in the study area during these secondary sighting occasions of the first primary interval.

As before, these data may be summarized into artificial individual encounter histories similar to those of the mark-recapture robust design. Now, both the number of marked animals in the super population ($\boldsymbol{T}$) and the total number of unmarked individuals seen ($\boldsymbol{T_u}$) during each secondary occasion must be entered after the encounter histories under the headings '**`Marked Superpopulation Group=1`**' and '**`Unmarked Seen Group=1`**' such that the resulting encounter history file would be:

```
/* No Individual Marks 1 Group */
/* 12 occasions, 3 primary, 4 secondary each */


/* Begin Input File */

111111111111        8;
111011111111        1;
110011011111        2;
110011010111        2;
110011000101        1;
110010000001        1;
100010000001        1;
100000000000        1;
000000000000        1;
00.000000000        1;
00.000000.00        1;
00.000.00.00        1;
00.000.00..0        1;
0..000.00..0        1;
0..00..00..0        4;
...00..00..0        1;
...0...00..0        1;
.......00..0        2;
.......0...0        1;
```

```
...........0        1;


Marked Superpopulation Group=1;
28 29 30 30 30 33 33 33 33 34 34 34;

Unmarked Seen Group=1;
264 161 152 217 217 160 195 159 166 152 175 190;

/* End Input File */
```

Notice the sums of the encounter history columns (when multiplied by the corresponding frequency) equal m_{ij}, and the sums of the non-missing entries (i.e., not '.') for each column equal n_{ij}. If these two conditions are satisfied, then the data have been correctly manipulated into artificial encounter histories.

With no individually identifiable marks, only the parameters p_{ij}, $\bar{N}_j$, α_{ij}, and N_j^* should be estimated, and σ_j needs to be fixed to zero. The analysis using the encounter history data contained in (`IELNE_NoIndividualMarks.inp`) yielded the following results for the fully time- and session-dependent model in **MARK**:

```
Real Function Parameters of {p(t*session) Nbar(session) alpha(t*session) Nstar(session)}
                                                    95% Confidence Interval
 Parameter                 Estimate       Standard Error      Lower           Upper
-------------------------  -------------  --------------  --------------  --------------
    1:p Session 1          0.5920048       0.0656170       0.4600336       0.7119197
    2:p Session 1          0.5336830       0.0849114       0.3696301       0.6907604
    3:p Session 1          0.5334205       0.0791715       0.3799858       0.6807828
    4:p Session 1          0.4660226       0.0528644       0.3652849       0.5696100
    5:sigma Session 1      0.0000000       0.0000000       0.0000000       0.0000000       Fixed
    6:Nbar Session 1       397.32353       37.779741       345.08949       499.99984
    7:alpha Session 1      77.488465       23.018541       32.372124       122.60481
    8:alpha Session 1      -67.485508      35.691739       -137.44132      2.4703031
    9:alpha Session 1      -95.435499      32.604481       -159.34028      -31.530716
   10:Nstar Session 1      494.93170       21.786836       460.75810       547.81022
   11:p Session 2          0.5299575       0.0619746       0.4091072       0.6473942
   12:p Session 2          0.5553905       0.0782237       0.4016923       0.6991743
   13:p Session 2          0.6222315       0.0715188       0.4756901       0.7493929
   14:p Session 2          0.3737708       0.0454746       0.2896954       0.4662306
   15:sigma Session 2      0.0000000       0.0000000       0.0000000       0.0000000       Fixed
   16:Nbar Session 2       385.24069       35.290100       331.28861       472.92637
   17:alpha Session 2      54.472940       22.564238       10.247032       98.698847
   18:alpha Session 2      -71.850164      29.657658       -129.97918      -13.721152
   19:alpha Session 2      -57.222270      25.778240       -107.74762      -6.6969177
   20:Nstar Session 2      475.22043       20.428033       443.38777       525.10066
   21:p Session 3          0.4143881       0.0495080       0.3217547       0.5134995
   22:p Session 3          0.7132910       0.0934712       0.5038977       0.8590295
   23:p Session 3          0.6569720       0.0793330       0.4899648       0.7924578
   24:p Session 3          0.4701110       0.0539388       0.3672353       0.5755906
   25:sigma Session 3      0.0000000       0.0000000       0.0000000       0.0000000       Fixed
   26:Nbar Session 3       346.12174       29.879666       300.03823       419.77196
   27:alpha Session 3      80.815452       21.159041       39.343731       122.28717
   28:alpha Session 3      -113.10159      28.082685       -168.14366      -58.059529
   29:alpha Session 3      -59.723073      25.125391       -108.96884      -10.477305
   30:Nstar Session 3      452.02721       20.950900       418.31832       501.69599
```

Here the mean population size using the study area during the first primary interval was $\hat{\bar{N}}_1 = 397.3$. The total population associated with the study area during the first primary interval was $\hat{N}_1^* = 494.9$.

The estimates for α suggest the population within the study area fluctuated, with $\hat{N}_{11} = \hat{\tilde{N}}_1 + \hat{\alpha}_{11} = 474.8$, $\hat{N}_{21} = \hat{\tilde{N}}_1 + \hat{\alpha}_{21} = 329.8$, $\hat{N}_{31} = \hat{\tilde{N}}_1 + \hat{\alpha}_{31} = 301.9$, and $\hat{N}_{31} = \hat{\tilde{N}}_1 - \sum_{i=1}^{k_1-1} \hat{\alpha}_{i1} = 482.8$.

Suppose temporary emigration from the study area during primary interval j is constant and completely random. In this case, the expected population size within the study area doesn't change despite the fact that individuals freely move in and out. Using the same data, this hypothesis may be explored by fixing $\alpha_{ij} = 0$ $(i = 1, \ldots, k_j - 1)$ in **MARK**:

Real Function Parameters of {p(t*session) N(session) Nstar(session)}

Parameter	Estimate	Standard Error	95% Confidence Interval Lower	Upper	
1:p Session 1	0.6439324	0.0636854	0.5120143	0.7571063	
2:p Session 1	0.4029707	0.0440082	0.3204665	0.4913568	
3:p Session 1	0.3684690	0.0411356	0.2920867	0.4520701	
4:p Session 1	0.5153590	0.0531874	0.4119457	0.6174735	
5:sigma Session 1	0.0000000	0.0000000	0.0000000	0.0000000	Fixed
6:Nbar Session 1	436.60462	40.188783	376.61052	538.64137	
7:alpha Session 1	0.0000000	0.0000000	0.0000000	0.0000000	Fixed
8:alpha Session 1	0.0000000	0.0000000	0.0000000	0.0000000	Fixed
9:alpha Session 1	0.0000000	0.0000000	0.0000000	0.0000000	Fixed
10:Nstar Session 1	532.11187	24.630622	494.78594	593.80284	
11:p Session 2	0.6078401	0.0608881	0.4844027	0.7188762	
12:p Session 2	0.4536115	0.0486381	0.3610701	0.5494736	
13:p Session 2	0.5320235	0.0548963	0.4246068	0.6365519	
14:p Session 2	0.4483689	0.0482075	0.3567993	0.5435784	
15:sigma Session 2	0.0000000	0.0000000	0.0000000	0.0000000	Fixed
16:Nbar Session 2	383.50015	34.946464	330.10908	470.38476	
17:alpha Session 2	0.0000000	0.0000000	0.0000000	0.0000000	Fixed
18:alpha Session 2	0.0000000	0.0000000	0.0000000	0.0000000	Fixed
19:alpha Session 2	0.0000000	0.0000000	0.0000000	0.0000000	Fixed
20:Nstar Session 2	480.30601	23.005205	444.82403	537.00999	
21:p Session 3	0.4922281	0.0511459	0.3936081	0.5914567	
22:p Session 3	0.4615808	0.0488057	0.3684460	0.5574769	
23:p Session 3	0.5228970	0.0535107	0.4185439	0.6252887	
24:p Session 3	0.5730778	0.0573259	0.4588866	0.6799768	
25:sigma Session 3	0.0000000	0.0000000	0.0000000	0.0000000	Fixed
26:Nbar Session 3	359.57710	31.935979	309.89200	437.67227	
27:alpha Session 3	0.0000000	0.0000000	0.0000000	0.0000000	Fixed
28:alpha Session 3	0.0000000	0.0000000	0.0000000	0.0000000	Fixed
29:alpha Session 3	0.0000000	0.0000000	0.0000000	0.0000000	Fixed
30:Nstar Session 3	466.75871	23.909342	429.24320	524.77071	

When fixing $\alpha_{ij} = 0$, the model may still be used to estimate both the super population size (N_j^*) and the population size within the study area $\bar{N}_j = N_{ij}$ $(i = 1, \ldots, k_j)$. For these data, however, the AIC_c evidence strongly favors the previous model ($\Delta AIC_c = 67.4$!).

18.3.2. Individually identifiable marks

As with the regular mixed logit-normal model with individually identifiable marks, the encounter histories are constructed with tk_j possible encounters representing δ_{sij} for individual s during secondary occasion i of primary interval j. If an individual is not yet marked or a marked individual is outside of the study area during secondary occasion i of primary interval j, then missing values (.) are included for that occasion in the encounter history. As before, the total number of marks seen but not identified to individual during secondary occasion i of primary interval j (ϵ_{ij}) are also entered into the encounter

history file. Using the same data from the previous example with one group and

$$
\begin{aligned}
t &= 3 \\
k_j &= 4 \\
\mathbf{n} &= \{27, 22, 18, 29, 28, 23, 20, 32, 31, 19, 21, 33\} \\
\mathbf{T} &= \{28, 29, 30, 30, 30, 33, 33, 33, 33, 34, 34, 34\} \\
\mathbf{m} &= \{17, 15, 9, 8, 16, 14, 9, 13, 11, 14, 13, 16\} \\
\boldsymbol{T}_u &= \{264, 161, 152, 217, 217, 160, 195, 159, 166, 152, 175, 190\} \\
\boldsymbol{\epsilon} &= (0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)
\end{aligned}
$$

one possible encounter history file incorporating individually identifiable marks would be:

```
/* Individual Marks 1 Group */
/* 12 occasions, 3 primary, 4 secondary each */
/* Marked individuals off the study area or not yet marked indicated by "." in encounter history */


/* Begin Input File */

11010..00100        1;
.........1.0        1;
11.110110110        1;
0011.1001..1        1;
...00.1001.1        1;
0.0000.0000.        1;
1.11111001.0        1;
0..111.11..1        1;
0110000011.0        1;
111011111.11        1;
1111110101.1        1;
1110..100.11        1;
00.000000000        1;
1.001..10.00        1;
111011.0..11        1;
11.01..01..1        1;
.00010011110        1;
.....11011.1        1;
01.01.000.01        1;
000011101010        1;
110.10.11111        1;
1..00.0011.0        1;
11.010010..0        1;
......000111        1;
.01001010.10        1;
11.011.10100        1;
11.101110.10        1;
011000.10.00        1;
00.001.00001        1;
100000.000.0        1;
0..00.10...1        1;
1.011..11.11        1;
```

```
110011.00.10        1;
.....10.0111        1;


Marked Superpopulation Group=1;
28 29 30 30 30 33 33 33 33 34 34 34;

Unmarked Seen Group=1;
264 161 152 217 217 160 195 159 166 152 175 190;

Marked Unidentified Group=1;
0 0 0 0 0 0 0 0 0 0 0 0;

/* End Input File */
```

Note that the sums of each column $\sum_{s=1}^{n_{ij}} \delta_{sij} = m_{ij} - \epsilon_{ij}$. The first encounter history describes a marked individual that was in the super population of marked individuals ($\boldsymbol{T}$) during all three primary intervals. This individual was outside the study area on the second and third secondary occasions of the second primary interval. The second encounter history from the top describes an individual that was not in the marked super population during the first and second primary intervals. This individual may not have been marked until sometime during the third primary interval or it may have already been marked but didn't use the study area during the first or second primary intervals. Either way, it's not included in T_{i1} or T_{i2}. We avoid needing to distinguish between these two possibilities in the encounter history by providing **MARK** with the known values for all T_{ij} under '**`Marked Superpopulation Group=1`**.'

Because marks are individually identifiable, individual heterogeneity models may be explored with these data. The analysis using these encounter history data (`IELNE_NoIndividualMarks.inp`) yielded the following results for the fully time- and session-dependent model in **MARK**:

```
Real Function Parameters of {p(t*session) sigma(session) Nbar(session) alpha(t*session) Nstar(session)}
                                                        95% Confidence Interval
 Parameter                 Estimate       Standard Error      Lower           Upper
-------------------------  -------------  --------------  --------------  --------------
    1:p Session 1          0.6133918       0.0910340       0.4278053       0.7710052
    2:p Session 1          0.5616208       0.1194894       0.3310362       0.7683446
    3:p Session 1          0.5413694       0.1033785       0.3429405       0.7274912
    4:p Session 1          0.4574051       0.0730364       0.3213478       0.6001278
    5:sigma Session 1      1.0205809       0.4924485       0.4163080       2.5019584
    6:Nbar Session 1       394.61098       44.452039       337.29368       522.89418
    7:alpha Session 1      79.083762       23.173848       33.663020       124.50450
    8:alpha Session 1      -73.737498      35.569234       -143.45320      -4.0217987
    9:alpha Session 1      -93.058635      32.321863       -156.40949      -29.707783
   10:Nstar Session 1      494.08538       22.358088       459.06840       548.42536
   11:p Session 2          0.5324475       0.0802870       0.3770380       0.6818062
   12:p Session 2          0.5597357       0.0973820       0.3694818       0.7339233
   13:p Session 2          0.6357481       0.0878476       0.4534913       0.7859170
   14:p Session 2          0.3512142       0.0600856       0.2440611       0.4758014
   15:sigma Session 2      0.9087787       0.4205942       0.3832277       2.1550599
   16:Nbar Session 2       387.54071       40.782499       327.25659       492.00581
   17:alpha Session 2      54.176870       22.768204       9.5511885       98.802551
   18:alpha Session 2      -71.817931      29.415765       -129.47283      -14.163031
   19:alpha Session 2      -56.822380      26.126235       -108.02980      -5.6149591
   20:Nstar Session 2      477.60634       21.044542       445.03053       529.30608
   21:p Session 3          0.3978764       0.0894500       0.2411951       0.5787139
   22:p Session 3          0.8554374       0.1033849       0.5347726       0.9682158
```

```
   23:p Session 3          0.7842213      0.1143219      0.4915720      0.9317948
   24:p Session 3          0.4882646      0.0998619      0.3035819      0.6762078
   25:sigma Session 3      1.7968113      0.6252497      0.9262340      3.4856536
   26:Nbar Session 3       329.55557      33.183455      281.69496      416.44233
   27:alpha Session 3      79.218362      20.744845      38.558465      119.87826
   28:alpha Session 3      -110.37310     25.613884      -160.57631     -60.169884
   29:alpha Session 3      -58.532733     22.476479      -102.58663     -14.478834
   30:Nstar Session 3      432.33206      21.148660      398.57525      482.84169
```

For the model ignoring individual heterogeneity:

```
Real Function Parameters of {p(t*session) Nbar(session) alpha(t*session) Nstar(session)}
                                                          95% Confidence Interval
 Parameter                 Estimate       Standard Error      Lower          Upper
-------------------------  -------------  --------------  -------------  -------------
    1:p Session 1          0.5920050      0.0656170      0.4600338      0.7119199
    2:p Session 1          0.5336829      0.0849114      0.3696300      0.6907603
    3:p Session 1          0.5334203      0.0791715      0.3799856      0.6807828
    4:p Session 1          0.4660228      0.0528644      0.3652850      0.5696102
    5:sigma Session 1      0.0000000      0.0000000      0.0000000      0.0000000      Fixed
    6:Nbar Session 1       397.32348      37.779732      345.08945      499.99977
    7:alpha Session 1      77.488323      23.018552      32.371960      122.60469
    8:alpha Session 1      -67.485442     35.691773      -137.44132     2.4704343
    9:alpha Session 1      -95.435342     32.604530      -159.34022     -31.530462
   10:Nstar Session 1      494.93154      21.786834      460.75796      547.81007
   11:p Session 2          0.5299575      0.0619746      0.4091071      0.6473944
   12:p Session 2          0.5553905      0.0782238      0.4016923      0.6991744
   13:p Session 2          0.6222314      0.0715188      0.4756901      0.7493929
   14:p Session 2          0.3737708      0.0454746      0.2896953      0.4662307
   15:sigma Session 2      0.0000000      0.0000000      0.0000000      0.0000000      Fixed
   16:Nbar Session 2       385.24068      35.290114      331.28859      472.92641
   17:alpha Session 2      54.472932      22.564274      10.246954      98.698911
   18:alpha Session 2      -71.850162     29.657677      -129.97921     -13.721115
   19:alpha Session 2      -57.222263     25.778259      -107.74765     -6.6968749
   20:Nstar Session 2      475.22042      20.428040      443.38775      525.10067
   21:p Session 3          0.4143880      0.0495079      0.3217546      0.5134994
   22:p Session 3          0.7132910      0.0934712      0.5038977      0.8590295
   23:p Session 3          0.6569718      0.0793330      0.4899648      0.7924576
   24:p Session 3          0.4701108      0.0539388      0.3672353      0.5755904
   25:sigma Session 3      0.0000000      0.0000000      0.0000000      0.0000000      Fixed
   26:Nbar Session 3       346.12180      29.879668      300.03828      419.77202
   27:alpha Session 3      80.815496      21.159034      39.343789      122.28720
   28:alpha Session 3      -113.10166     28.082677      -168.14371     -58.059612
   29:alpha Session 3      -59.723082     25.125383      -108.96883     -10.477330
   30:Nstar Session 3      452.02731      20.950896      418.31842      501.69607
```

The interpretation of the parameters remains the same as before. In this case, AIC_c lends more support to the model including individual heterogeneity ($\Delta AIC_c = 7.0$). Notice that because all $\epsilon_{ij} = 0$ for these data, the estimates from the no-heterogeneity model with individually identifiable marks are the same as those for the same model when there were no individually identifiable marks.

18.4. The Poisson-log normal mark-resight model

For use when the number of marked individuals in the population may be unknown or sampling is with replacement within secondary sampling occasions (or there is no concept of a distinct secondary sampling occasion without replacement). Marks must be individually identifiable. See McClintock *et al.* 2009a and McClintock & White 2009 for full details.

Data:

t = the number of primary sampling intervals (through time, groups, or time and groups)

n_j = the exact number of marked individuals in the population during primary interval j

n_j^* = total number of marked individuals resighted at least once and known to be in the population

c_j = total number of individuals captured (e.g., for marking) immediately prior to primary interval j and therefore assumed to be present in the population during primary interval j, but not resighted during primary interval j

c_j^* = $n_j^* + c_j$ = total number of marked individuals captured immediately prior to primary interval j or resighted at least once during primary interval j. When the number of marks is known exactly, $c_j^* = n_j$. When the number of marks is unknown this is the minimum number of marked individuals known to be in the population

y_{sj} = Poisson random variable for the total number of times individual s was resighted during primary interval j

ϵ_j = total number of times an individual was sighted and identified as marked, but not identified to individual identity during primary interval j

T_{uj} = total unmarked individual sightings during primary interval j

Parameters:

U_j = number of unmarked individuals in the population during primary interval j

α_j = intercept (on log scale) for mean resighting rate during primary interval j. If there is no individual heterogeneity ($\sigma_j = 0$), once back-transformed from the log scale the real parameter estimate can be interpreted as the mean resighting rate for the entire population

σ_j^2 = individual heterogeneity level (on the log scale) during primary interval j, i.e., the additional variance due to a random individual heterogeneity effect with mean zero

φ_j = apparent survival between primary intervals j and $j+1$, $j = \{1, ..., t-1\}$

γ_j'' = probability of transitioning from an observable state at time j (e.g., on the study area) to an unobservable state at time $j+1$ (e.g., off the study area), $j = \{1, ..., t-1\}$. This is equivalent to transition probability ψ_j^{OU} of Kendall & Nichols (2002)

γ_j' = probability of remaining at an unobservable state at time $j+1$ (e.g., off the study area) when at an unobservable state at time j, $j = \{2, ..., t-1\}$. This is equivalent to $1 - \psi_j^{UO}$ of Kendall & Nichols (2002)

Derived Parameters:

λ_j = overall mean resighting rate for primary occasion j. This is a parameter derived as a function of α_j, σ_j^2, and ϵ_j. Note that when $\sigma_j = 0$ and $\epsilon_j = 0$, then the real parameter estimate for α_j is identical to the derived parameter estimate for λ_j

N_j = $U_j + n_j$ = total population size during primary occasion j. This is a derived parameter because **MARK** actually estimates U_j in the model. If n_j is unknown, then N_j is derived as $U_j + n_j^*/[1 - \exp(-\lambda_j)]$, where $n_j^*/[1 - \exp(-\lambda_j)]$ is the number of marked individuals

18.4.1. Closed resightings only

If interest is only in abundance estimates for different groups (or t primary intervals for group(s) with few or no marked individuals in common across the intervals), then the mark-resight Poisson-log normal model may be used in a fashion analogous to the closed mark-recapture models of Otis *et al.* (1987). In contrast to the closed mark-recapture models of Otis *et al.* (1987), individual covariates may be used in modeling resighting probabilities. However, **because the data consist of the total number of times each marked individual was resighted, the encounter histories must be modified to reflect this different type of encounter data**. If the number of marks is known exactly, then n_j, y_{sj}, ϵ_j and T_{uj} are the same data used for Bowden's estimator (Bowden & Kufeld 1995) in **NOREMARK** (White 1996), but the Poisson-log normal model will generally be more efficient because information about resighting probabilities may be borrowed across time or groups (McClintock *et al.* 2009a).

The number of marks available for each of the groups or t primary intervals may be known or unknown. The encounter history file contains individual encounter histories composed of the y_{sj} resightings, the frequencies and group(s) to which each encounter history pertains, the T_{uj} unmarked sightings and group(s) to which they pertain, the ϵ_j unidentified marks and the group(s) to which they pertain, and whether or not the number of marks is known exactly for each group. **Instead of the familiar 0's and 1's of other MARK encounter histories, these histories simply contain the y_{sj} for each marked individual** s. Two character spaces are allocated to allow $y_{sj} > 9$. Note that this coding does not allow $y_{sj} > 99$. For reasons that will become clear in the next section covering the robust design Poisson-log normal model, **entries for which $y_{sj} = 0$ are entered using '+0' instead of '00'**. Further, (unlike the logit-normal model and mark-recapture robust design), because the Poisson-log normal model does not condition on distinct secondary resighting occasions, **the number of encounter occasions entered into MARK when creating a new analysis is the number of primary occasions**.

For instance, suppose in a very simple example that there were two groups and $t = 1$ primary interval with known $n_1 = 3$, $y_{11} = 2$, $y_{21} = 3$, $y_{31} = 0$, $T_{u_1} = 11$, and $\epsilon_1 = 2$ for the first group, and $n_1 = 3$, $y_{11} = 0$, $y_{21} = 0$, $y_{31} = 12$, $T_{u_1} = 5$, and $\epsilon_1 = 3$ for the second group. The encounter history file would be:

```
/* Poisson log-normal mark-resight */
/* Occasions=1 groups=2 */

/* Begin Input File */

02 1 0;
03 1 0;
+0 1 0;
+0 0 1;
+0 0 1;
12 0 1;

Unmarked Seen Group=1;
11;

Unmarked Seen Group=2;
5;

Marked Unidentified Group=1;
2;

Marked Unidentified Group=2;
```

```
3;

Known Marks Group=1;
3;

Known Marks Group=2;
3;

/* End Input File */
```

The columns following the encounter histories are the frequencies for the two groups, just as would be done in other **MARK** encounter history files. Under '**Unmarked Seen**', the T_{uj} are entered separately for each group. The '**Marked Unidentified**' data (ϵ_j) are entered in the same fashion separately for each group. Similarly, the '**Known Marks**' headings contain the n_j for each group.

Using the same example, but now with the number of marks being unknown for the second group, the encounter history file must be modified to reflect that n_2 is unknown and $y_{s2} = 0$ is no longer observed:

```
/* Poisson log-normal mark-resight */
/* occasions=1 groups=2 */

/* Begin Input File */

02 1 0;
03 1 0;
+0 1 0;
12 0 1;

Unmarked Seen Group=1;
11;

Unmarked Seen Group=2;
5;

Marked Unidentified Group=1;
2;

Marked Unidentified Group=2;
3;

Known Marks Group=1;
3;

Known Marks Group=2;
0;

/* End Input File */
```

Here, the encounter histories for $y_{12} = 0$ and $y_{22} = 0$ have been removed because they cannot be observed if the number of marked individuals in the population (n_2) is unknown. Further, under '`Known Marks;`' there is now a '`0`' for the second group. **By including a '`0`' for the second group's '`Known Marks`',**

MARK knows the number of marks is unknown and will use the zero-truncated Poisson-log normal model.

It is possible that the number of marks may be unknown for a given group, but some marking was conducted immediately prior to the primary sampling interval of interest. Here, some additional information is known about the minimum number of marks in the population because those (previously marked or newly marked) individuals captured during the marking period are known to have been present and available for resighting (even if they were not resighted during the interval of interest). Suppose this were the case in the above example, such that the second individual of the second group was captured and marked immediately prior to resighting surveys but never resighted. This information (although not used in the zero-truncated likelihood) may be included in the encounter history file to make the lower bound for $N_2 \geq c_2^*$:

```
/* Poisson log-normal mark-resight */
/* occasions=1 groups=2 */

/* Begin Input File */

02 1 0;
03 1 0;
+0 1 0;
+0 0 1;
12 0 1;

Unmarked Seen Group=1;
11;

Unmarked Seen Group=2;
5;

Marked Unidentified Group=1;
2;

Marked Unidentified Group=2;
3;

Known Marks Group=1;
3;

Known Marks Group=2;
0;

/* End Input File */
```

Because the '**Known Marks;**' is still '0' for the second group, **MARK** knows the actual number of marks is unknown and to use the zero-truncated model for the second group, but $c_2^* = 2$ (instead of $n_2^* = 1$) will be used in establishing the lower bound for N_2. When the number of marks is unknown, the information provided by such encounters via capture events will become more useful when considering the robust design Poisson-log normal model in the next section.

Now to analyze a more realistic data set where the number of marks was known for the first group but not for the second. No marking occurred immediately prior to resighting surveys for the second group, so $c_2^* = n_2^*$, and therefore no '+0' encounter histories are included for the second group. For the

first group, $n_1 = 60$, $T_{u_1} = 1237$, and $\epsilon_1 = 10$. For the second group, $n_1^* = 33$, $T_{u_1} = 588$,and $\epsilon_1 = 5$:

```
/* Poisson log-normal mark-resight */
/* Occasions=1 groups=2 */

/* Begin Input File */

02 1 0;
03 1 0;
03 1 0;
01 1 0;
01 1 0;
01 1 0;
02 1 0;
09 1 0;
05 1 0;
01 1 0;
01 1 0;
01 1 0;
03 1 0;
03 1 0;
02 1 0;
06 1 0;
04 1 0;
02 1 0;
03 1 0;
01 1 0;
02 1 0;
01 1 0;
03 1 0;
04 1 0;
03 1 0;
03 1 0;
05 1 0;
03 1 0;
04 1 0;
04 1 0;
+0 1 0;
04 1 0;
01 1 0;
03 1 0;
02 1 0;
01 1 0;
03 1 0;
02 1 0;
03 1 0;
05 1 0;
06 1 0;
03 1 0;
01 1 0;
04 1 0;
07 1 0;
03 1 0;
```

```
+0 1 0;
06 1 0;
+0 1 0;
04 1 0;
+0 1 0;
02 1 0;
02 1 0;
02 1 0;
02 1 0;
05 1 0;
02 1 0;
01 1 0;
04 1 0;
+0 1 0;
02 0 1;
02 0 1;
04 0 1;
01 0 1;
02 0 1;
01 0 1;
01 0 1;
01 0 1;
04 0 1;
03 0 1;
01 0 1;
05 0 1;
02 0 1;
02 0 1;
05 0 1;
02 0 1;
01 0 1;
05 0 1;
01 0 1;
02 0 1;
07 0 1;
01 0 1;
03 0 1;
05 0 1;
03 0 1;
03 0 1;
04 0 1;
02 0 1;
03 0 1;
05 0 1;
02 0 1;
02 0 1;
02 0 1;

Unmarked Seen Group=1;
1237;

Unmarked Seen Group=2;
588;
```

```
Marked Unidentified Group=1;
10;

Marked Unidentified Group=2;
5;

Known Marks Group=1;
60;

Known Marks Group=2;
0;

/* End Input File */
```

The analysis for these data (`Poisson_TwoGroups.inp`) yielded the following results for the most general model:

```
                Real Function Parameters of {alpha(g)sigma(g)U(g)}

                                                              95% Confidence Interval
Parameter                 Estimate       Standard Error       Lower           Upper
------------------------  -------------  -------------  -------------  -------------
   1:alpha                2.6274189      0.2483643      2.1839589      3.1609248
   2:alpha                2.3834952      0.3632005      1.7711208      3.2076012
   3:sigma                0.2782579      0.1405534      0.1093112      0.7083213
   4:sigma                0.2316744      0.2787288      0.0362715      1.4797580
   5:U                    426.66770      37.155745      359.83441      505.91416
   6:U                    227.09486      29.801418      175.78405      293.38314
```

In most situations, these real parameter estimates may not be of interest. The derived parameters for abundance (N) and mean resighting rate (λ) are typically what we want:

```
                  Estimates of Derived Parameters
            Population Estimates of {alpha(g)sigma(g)U(g)}

                                                 95% Confidence Interval
Grp. Occ.   N-hat           Standard Error       Lower           Upper
---- ----   --------------  --------------  --------------  --------------
  1     1   486.66770       37.155822       419.12029       565.10136
  2     1   263.21721       30.821410       209.40169       330.86314

       Mean Resighting Rate Estimates of {alpha(g)sigma(g)U(g)}

                                                 95% Confidence Interval
Grp. Occ.   Lambda-hat      Standard Error       Lower           Upper
---- ----   --------------  --------------  --------------  --------------
  1     1   2.8977973       0.2355306       2.4716992       3.3973507
  2     1   2.5867444       0.3200561       2.0315747       3.2936257
```

Here are the results for the model with no group effects on α_j or σ_j:

Real Function Parameters of {alpha(.)sigma(.)U(g)}

Parameter	Estimate	Standard Error	95% Confidence Interval Lower	Upper
1:alpha	2.5449662	0.2037816	2.1758646	2.9766800
2:sigma	0.2670036	0.1248112	0.1117130	0.6381611
3:U	440.94680	32.590191	381.55642	509.58148
4:U	211.45044	17.316388	180.14242	248.19966

Estimates of Derived Parameters
Population Estimates of {alpha(.)sigma(.)U(g)}

Grp.	Occ.	N-hat	Standard Error	95% Confidence Interval Lower	Upper
1	1	500.94680	32.590259	441.03409	568.99842
2	1	246.99366	17.749865	214.58185	284.30115

Mean Resighting Rate Estimates of {alpha(.)sigma(.)U(g)}

Grp.	Occ.	Lambda-hat	Standard Error	95% Confidence Interval Lower	Upper
1	1	2.8039855	0.1882162	2.4586823	3.1977840
2	1	2.7779927	0.1902567	2.4294158	3.1765839

Here are the results for the model with no group effect on α_j and $\sigma_j = 0$:

Real Function Parameters of {alpha(.)sigma(.)=0 U(g)}

Parameter	Estimate	Standard Error	95% Confidence Interval Lower	Upper	
1:alpha	2.6488895	0.1731506	2.3306735	3.0105529	
2:sigma	0.0000000	0.0000000	0.0000000	0.0000000	Fixed
3:U	439.16724	29.754643	384.61298	501.45959	
4:U	210.59709	15.810414	181.81833	243.93104	

Estimates of Derived Parameters
Population Estimates of {alpha(.)sigma(.)=0 U(g)}

Grp.	Occ.	N-hat	Standard Error	95% Confidence Interval Lower	Upper
1	1	499.16724	29.754705	444.17194	560.97181
2	1	246.10883	16.203557	216.34382	279.96896

```
      Mean Resighting Rate Estimates of {alpha(.)sigma(.)=0 U(g)}

                                                95% Confidence Interval
Grp. Occ.   Lambda-hat      Standard Error      Lower           Upper
---- ----   --------------  --------------  --------------  --------------
  1     1   2.8155562       0.1731506       2.4961207       3.1758707
  2     1   2.7896881       0.1750062       2.4672233       3.1542988
```

Note that to run models without individual heterogeneity, σ_j must be fixed to zero. When $\sigma = 0$, the real parameter estimate of α may be interpreted as the overall mean resighting rate ignoring unidentified marks, but λ is an overall mean resighting rate that takes unidentified marks into account.

18.4.2. Full-likelihood robust design

If interest is in apparent survival (φ), transition probabilities between observable and unobservable states (γ' and γ''), and abundance (N) for one or more groups through time, then a mark-resight robust design analogous to the mark-recapture robust design of Kendall, Pollock & Brownie (1995) and Kendall, Nichols & Hines (1997) may be employed. Full details on the model may be found in McClintock & White (2009). In contrast to the modeling of recapture probabilities in the mark-recapture robust design utilizing the closed capture models of Otis *et al.* (1987), the mark-resight robust design may incorporate individual covariates in modeling resighting probabilities.

The encounter history files are similar to those from the previous Closed Resightings model, but now the open period encounter process for individuals with permanent field-readable marks may be modeled through time across t primary sampling intervals in a robust design. For instance, if an individual s was encountered $y_{s1} = 4$ times during the first primary interval and $y_{s2} = 2$ times during the second primary interval, then the encounter history would be '`0402`'. Each encounter history will contain $2t$ characters, again allowing two characters for each y_{sj}. Because the number of marks can be known or unknown for any given primary interval, the primary intervals must again be identified as such under the '`Known Marks`' heading in the encounter history file. In the individual encounter histories, **a '`+0`' indicates that the individual was known to be a marked individual available for resighting during primary interval j but never resighted**. Therefore, when the number of marks is unknown, the total number of '`+0`' entries during primary interval j is equal to c_j as defined above. **A '`-0`' indicates a previously encountered individual that was not encountered (via capture OR resighting) during primary interval j, and only applies when the number of marks is unknown (i.e., when the number of marks is known a '`-0`' is impossible)**. Lastly, a '..' indicates a marked individual who has not yet been encountered prior to and during primary interval j OR an individual that is known to no longer be in the marked population (due to removal, mortality, or permanent emigration) during and after primary interval j. As in the regular CJS model in **MARK**, any '..' contributes no information to the estimation of parameters. When n_j is known, '`+0`' contributes information towards estimation of survival, transition probabilities, resighting probabilities, and abundance. When n_j is unknown, '`+0`' contributes information towards estimating survival and transition probabilities, but makes no contribution to the estimation of resighting probabilities or abundance (but it does affect the minimum lower bound for N_j as described in the previous section). A '`-0`' contributes no information to the estimation of resighting probabilities or abundance (it is only a valid entry when the number of marks is unknown), and is equivalent to a '`0`' in the regular CJS encounter history for **MARK**. It therefore only contributes to the estimation of survival and transition probabilities. As before, the encounter histories are followed by group frequencies in the usual **MARK** encounter history file. The entries for '**`Unmarked Seen`**', '**`Marked Unidentified`**', and '**`Known Marks`**' are the same as described earlier and are

entered separately for each group.

In the following example encounter history file with a single group and $t = 4$ primary intervals, the number of marks are known for the first and second primary intervals, but unknown for the third and fourth. **Because the model does not condition on distinct secondary resighting occasions, the number of encounters that are input into MARK is equal to the number of primary occasions** ($t = 4$ in this case). Capturing for marking occurred immediately prior to the first, second, and third occasion, but not the fourth occasion, so $n_4^* = c_4^*$. Here, $n_1 = 45$, $T_{u_1} = 1380$, $\epsilon_1 = 8$, $n_2 = 67$, $T_{u_2} = 1120$, $\epsilon_2 = 10$, $n_3^* = 56$, $T_{u_3} = 1041$, $\epsilon_3 = 9$, $n_4^* = 52$, $T_{u_4} = 948$, and $\epsilon_4 = 11$:

```
/* Poisson log-normal Mark-resight */
/* 4 occasions, 1 group */

/* Begin Input File */

....+002 1;
..06-0-0 1;
04060202 1;
+0010402 1;
070602-0 1;
04020606 1;
..020101 1;
060602-0 1;
..04-004 1;
040401-0 1;
03010103 1;
02030503 1;
..03+0-0 1;
070503-0 1;
04+00104 1;
01010401 1;
06060103 1;
02010602 1;
..0403-0 1;
..020306 1;
020202-0 1;
..050201 1;
02010103 1;
031002-0 1;
+0+00704 1;
01030102 1;
01010302 1;
..02-0-0 1;
..020210 1;
020301-0 1;
02+00503 1;
02+0+0-0 1;
02020302 1;
..080201 1;
..040603 1;
030304-0 1;
02020202 1;
..030107 1;
```

```
04050402 1;
+0050101 1;
..030605 1;
05+00101 1;
..04-003 1;
06020204 1;
..03-004 1;
..010201 1;
04+00303 1;
04040204 1;
01+00201 1;
0403-004 1;
01+00103 1;
..020307 1;
01060701 1;
..040101 1;
03040301 1;
..0404-0 1;
03050101 1;
05040202 1;
03010202 1;
05+00302 1;
01020202 1;
01+0+0-0 1;
01070202 1;
..050105 1;
02040205 1;
02010301 1;
..03-010 1;
..01+0-0 1;

Unmarked Seen Group=1;
1380 1120 1041 948;

Marked Unidentified Group=1;
8 10 9 11;

Known Marks Group=1;
45 67 0 0;

/* End Input File */
```

The first encounter history indicates this individual was not captured for marking until immediately prior to the third primary occasion, and the '+0' for the third occasion indicates that it was not resighted (although known to be a marked individual available for resighting during this occasion). This individual was then resighted twice during the fourth occasion. The second encounter history from the top indicates that this individual was only known to be marked and in the population during the second primary occasion (when it was resighted 6 times). Because the number of marks is known for the first primary interval, this individual must have been marked between the first and second primary intervals. As indicated by '-0', this individual was never encountered again when the number of marks was unknown during the third and fourth primary intervals. The third encounter history from the top indicates an individual who was known to be marked and available for resighting for all $t = 4$

occasions. The '+0' entry for the first primary occasion indicates that it was known to be marked and available for resighting, but never resighted. This individual was then resighted one, four, and two times during the second, third, and fourth intervals, respectively. The final encounter history describes an individual that was not marked until immediately prior to the second primary occasion, and during the second occasion it was resighted one time. It was then captured immediately prior to (but never resighted during) the third occasion. Because the number of marks was unknown for the third occasion, this '+0' primarily contributes information to the estimation of survival and transition probabilities (as described in the previous section). As indicated by '-0' this individual was then never resighted during the fourth occasion (and could not have been captured immediately prior to the occasion because no capturing took place). Because no individuals were captured (e.g., for marking) immediately prior to the fourth occasion (and the number of marked individuals was unknown), no '+0' appears in the entries for this occasion. Because no marked individuals were known to have left the population (due to removal, mortality, or permanent emigration), no '..' entries appear after an individual's first encounter. The '**Unmarked Seen**' entry tells **MARK** that 1380 unmarked sightings occurred during the first primary interval, 1120 during the second, 1041 during the third, and 948 during the fourth. The '**Marked Unidentified**' entry follows the same pattern. The '**Known Marks**' entry tells **MARK** that n_j is known for the first and second primary intervals ($n_1 = 46$, $n_2 = 60$), but unknown for the third and fourth (as indicated by '0' for these occasions).

As a simple two group example, suppose for the first group that $n_1 = 10$, $T_{u_1} = 800$, $\epsilon_1 = 4$, $n_2 = 14$, $T_{u_2} = 950$, $\epsilon_2 = 2$, $n_3^* = 11$, $T_{u_3} = 500$, $\epsilon_3 = 6$, $n_4^* = 8$, $T_{u_4} = 1201$, and $\epsilon_4 = 3$. For the second group, $n_1 = 11$, $T_{u_1} = 459$, $\epsilon_1 = 2$, $n_2^* = 14$, $T_{u_2} = 782$, $\epsilon_2 = 5$, $n_3^* = 15$, $T_{u_3} = 256$, $\epsilon_3 = 0$, $n_4^* = 11$, $T_{u_4} = 921$, and $\epsilon_4 = 1$. With capturing (e.g., for marking) occurring for both groups immediately prior to the first and second occasions, a possible encounter history file would be:

```
/* Poisson log-normal Mark-resight */
/* 4 occasions, 2 groups */

/* Begin Input File */
04060202 1 0;
..06-0-0 1 0;
+0010402 1 0;
070602-0 1 0;
04020606 1 0;
..020101 1 0;
060602-0 1 0;
..04-004 1 0;
040401-0 1 0;
03010103 1 0;
02030503 1 0;
..03-0-0 1 0;
070503-0 1 0;
04+00104 1 0;
01010401 0 1;
06060103 0 1;
02010602 0 1;
..0403-0 0 1;
..020306 0 1;
020202-0 0 1;
..050201 0 1;
02010103 0 1;
031002-0 0 1;
```

```
+0-00704 0 1;
01030102 0 1;
01010302 0 1;
..02-0-0 0 1;
..020210 0 1;
020301-0 0 1;
02+00503 0 1;

Unmarked Seen Group=1;
800 950 500 1201;

Unmarked Seen Group=2;
459 782 256 921;

Marked Unidentified Group=1;
4 2 6 3;

Marked Unidentified Group=2;
2 5 0 1;

Known Marks Group=1;
10 14 0 0;

Known Marks Group=2;
11 0 0 0;

/* End Input File */
```

Here, the encounter histories are followed by two columns for group frequencies in the usual **MARK** encounter history file manner. The entries for '**Unmarked Seen**', '**Marked Unidentified**', and '**Known Marks**' are entered separately for each group. The entries under '**Known Marks**' tell **MARK** that the number of marks was known for the first and second primary occasions of the first group ($n_1 = 10$, $n_2 = 14$) and for only the first primary occasion of the second group ($n_1 = 11$). Again, no '-0' can appear for a primary occasion where the number of marks is unknown. Notice that a '+0' appears in the encounter history for the last individual of the second group, but that the number of marks for this primary occasion was unknown. This indicates that this individual happened to be caught (e.g., during marking) immediately prior to the second primary occasion, but was never resighted. Hence, for the second group during the second primary interval , $n_2^* = 14$ and $c_2^* = 15$.

An analysis using the single group data (`Poisson_RobustDesign_OneGroup.inp`) yielded the following results for the random emigration model $\{\varphi(.)\gamma''(.) = \gamma'(.)\alpha(t)\ \sigma(t)U(t)\}$:

```
      Real Function Parameters of {phi(.) gamma'(.)=gamma''(.) alpha(t) sigma(t) U(t)}

                                                          95% Confidence Interval
Parameter                 Estimate       Standard Error      Lower           Upper
------------------------  -------------  --------------  --------------  --------------
   1:alpha                2.7638408       0.2886637       2.2534628       3.3898122
   2:alpha                2.6470841       0.2695821       2.1692136       3.2302279
   3:alpha                2.1173163       0.2745082       1.6439392       2.7270036
   4:alpha                2.1254054       0.3281373       1.5732477       2.8713520
   5:sigma                0.2368147       0.1786795       0.0635331       0.8827081
   6:sigma                0.4564778       0.1114859       0.2847935       0.7316598
```

```
   7:sigma                0.3925358       0.1552277       0.1859589       0.8285937
   8:sigma                0.5348317       0.1257812       0.3394039       0.8427864
   9:U                    456.73003       43.067154       379.81489       549.22102
  10:U                    362.54432       34.740271       300.59433       437.26168
  11:U                    427.89101       45.664583       347.33475       527.13045
  12:U                    358.01017       44.974968       280.14293       457.52102
  13:Phi                  0.9857548       0.0182401       0.8443633       0.9988683
  14:Gamma'              0.0552683       0.0363436       0.0147309       0.1862693

                        Estimates of Derived Parameters
Population Estimates of {phi(.) gamma'(.)=gamma'(.) alpha(.) sigma(.) U(t)}

                                                  95% Confidence Interval
Grp. Occ.   N-hat           Standard Error      Lower           Upper
---- ----   --------------  --------------  --------------  --------------
  1     1   524.94217       28.178946       472.55342       583.13891
  1     2   460.37288       24.092419       415.52193       510.06500
  1     3   425.58023       23.324431       382.26492       473.80369
  1     4   383.16101       21.077938       344.02552       426.74845

Mean Resighting Rate Estimates of {phi(.) gamma'(.)=gamma'(.) alpha(.) sigma(.) U(t)}

                                                  95% Confidence Interval
Grp. Occ.   Lambda-hat      Standard Error      Lower           Upper
---- ----   --------------  --------------  --------------  --------------
  1     1   2.8737886       0.1396905       2.6127801       3.1608711
  1     2   2.8452646       0.1396905       2.5843816       3.1324827
  1     3   2.8458811       0.1412053       2.5823048       3.1363607
  1     4   2.8932760       0.1416843       2.6286365       3.1845582
```

For model $\{\varphi(.)\gamma''(.) = \gamma'(.)\alpha(.)\sigma(.)U(t)\}$:

```
      Real Function Parameters of {Phi(.) gamma'(.)=gamma'(.) alpha(.) sigma(.) U(t)}

                                                          95% Confidence Interval
Parameter                 Estimate        Standard Error      Lower           Upper
------------------------  --------------  --------------  --------------  --------------
   1:alpha                2.4536985       0.1478956       2.1805245       2.7610956
   2:sigma                0.4376083       0.0655452       0.3268107       0.5859693
   3:N                    524.49384       28.499239       471.81002       583.68075
   4:N                    460.04989       24.370049       415.11342       510.78703
   5:N                    426.24093       23.102678       383.69402       474.39761
   6:N                    379.16926       20.875980       340.74421       422.70778
   7:Phi                  0.9858690       0.0178497       0.8499082       0.9988380
   8:Gamma'               0.0751540       0.0287552       0.0348592       0.1545672

                     Estimates of Derived Parameters
Population Estimates of {phi(.) gamma'(.)=gamma'(.) alpha(.) sigma(.) U(t)}

                                                  95% Confidence Interval
Grp. Occ.   N-hat           Standard Error      Lower           Upper
---- ----   --------------  --------------  --------------  --------------
```

```
  1     1   524.94217       28.178946       472.55342       583.13891
  1     2   460.37288       24.092419       415.52193       510.06500
  1     3   425.58023       23.324431       382.26492       473.80369
  1     4   383.16101       21.077938       344.02552       426.74845

Mean Resighting Rate Estimates of {phi(.) gamma'(.)=gamma'(.) alpha(.) sigma(.) U(t)}

                                                95% Confidence Interval
Grp. Occ.   Lambda-hat      Standard Error      Lower           Upper
---- ----   --------------  --------------  --------------  --------------
  1     1   2.8737886       0.1396905       2.6127801       3.1608711
  1     2   2.8452646       0.1396905       2.5843816       3.1324827
  1     3   2.8458811       0.1412053       2.5823048       3.1363607
  1     4   2.8932760       0.1416843       2.6286365       3.1845582
```

Here, AIC_c indicates much more support for the simpler model (1042.3 versus 1050.0). Notice that a significant population decline would be inferred from the latter model (but not the former), one of the advantages of borrowing information across primary intervals that the Poisson-log normal model provides over other previously available mark-resight estimators.

18.5. Which mark-resight model? Decision table

As previously described, there a variety of mark-resight models available to you in **MARK**: (1) the logit-normal estimator (*LNE*); (2) the immigration-emigration log-normal estimator (*IELNE*); and (3) the (zero-truncated) Poisson log-normal estimator [(*Z*)*PNE*].

The following summary table provides some guidance by comparing several of the important differences in the underlying assumptions:

estimator	*geographic closure*	*sampling with replacement*	*number of known marks*	*individually identifiable marks*
LNE	required	not allowed	required	not required
IELNE	not required	not allowed	required	not required
(*Z*)*PNE*	required	allowed	not required	required

Geographic closure is only required for *LNE* and (*Z*)*PNE* within primary sampling intervals. As described at the end of Section 18.1, closure assumptions may often be relaxed, but abundance estimates should be carefully interpreted under these circumstances.

18.6. Suggestions for mark-resight analyses in MARK

1. To start an analysis from scratch (after an encounter history file has been created), select the '**Mark-Resight**' data type. The option will then be given to select among '**Logit-Normal**', '**Immigration-Emigration Logit-Normal**,' or '**Poisson-log normal**'. For both '**Logit-Normal**'

and '**Immigration-Emigration Logit-Normal**' one doesn't specify whether or not individual marks were used. This is left to the user to keep track of (by not running any individual heterogeneity models if marks were not individually identifiable). For '**Poisson-log normal**' one doesn't need to specify robust design or not. If there are multiple primary occasions for the group(s), then **MARK** will automatically set up an analysis that includes the open period parameters (φ, γ'', and γ').

2. Because convergence with these models is sensitive to the starting values (particularly for N and σ), initial values (on the log scale) should always be manually provided in the '**Run**' window when using the design matrix. This means that if $N = 100$ and $\sigma = 0.5$, then $\log(N) = 4.6$ and $\log(\sigma) = -0.69$ should be provided as initial values. **MARK** provides its own initial values that usually work when running a model from the PIMs, so we suggest that an analysis begin with simple PIM models from which the initial values may then be provided for running more complex models and for when utilizing the design matrix. If convergence issues remain after following this strategy, we suggest trying a series of initial values covering the suspected range of the parameter(s) and possibly other '**Run window**' options such as '**Use Alt. Opt. Method**' or '**Do not standardize design matrix**'. It is typically fairly obvious when N does not converge correctly ('garbage' estimates, SE, or AIC_c), but it can be more tricky with σ. Sometimes the regular **MARK** optimization method can converge to a local maximum where $\hat{\sigma}$ is almost zero. Caution should be taken before concluding that such an estimate is reliable.
3. Even when using the sin link from the PIMs, **MARK** will sometimes get the parameter count wrong for the α parameters in the immigration-emigration logit-normal model. Extra care should be taken when using the model to verify the number of estimable parameters (e.g., for AIC_c calculation) is correct. We hope to have this issue resolved in the future.
4. The σ parameter must be fixed to zero in the '**Run**' window to examine a model that ignores individual heterogeneity in resighting probabilities.
5. When using the (immigration-emigration) logit-normal model, **MARK** by default assigns the log link to σ and N, and applies whatever link is specified in the Run window to p.
6. When using the Poisson model, **MARK** by default assigns the log link to α, σ, and N, and applies whatever link is specified in the '**Run**' window to φ, γ'', and γ' (if using the robust design).

18.7. References

Arnason, A. N., Schwarz, C. J., and Gerrard, J. M. 1991. Estimating closed population size and number of marked animals from sighting data. *Journal of Wildlife Management*, **55**, 716-730.

Bartmann, R. M., White, G. C., Carpenter, L. H., and Garrott, R. A. 1987. Aerial mark-recapture estimates of confined mule deer in pinyon-juniper woodland. *Journal of Wildlife Management*, **51**, 41-46.

Bowden, D. C., and Kufeld, R. C. 1995. Generalized mark-resight population size estimation applied to Colorado moose. *Journal of Wildlife Management*, **59**, 840-851.

Burnham, K. P., and Anderson, D. R. 2002. *Model Selection and Multi- model Inference: A Practical Information-Theoretic Approach*. 2nd edition. Springer-Verlag, New York, USA.

Kendall, W. L., and Nichols, J. D. 2002. Estimating state-transition probabilities for unobservable states using capture-recapture/resighting data. *Ecology*, **83**, 3276-3284.

Kendall, W. L., Nichols, J. D., and Hines, J. E. 1997. Estimating temporary emigration using capture-recapture data with Pollock's robust design. *Ecology*, **78**, 563-578.

Kendall, W. L., Pollock, K. H., and Brownie, C. 1995. A likelihood-based approach to capture-recapture estimation of demographic parameters under the robust design. *Biometrics*, **51**, 293-308.

McClintock, B. T., and White, G. C. 2009. A less field-intensive robust design for estimating demographic parameters with mark-resight data. *Ecology*, **90**, 313-320.

McClintock, B. T., White, G. C., Antolin, M. F., and Tripp, D. W. 2009a. Estimating abundance using mark-resight when sampling is with replacement or the number of marked individuals is unknown. *Biometrics*, **65**, 237-246.

McClintock, B. T., White, G. C., Burnham, K. P., and Pryde, M. A. 2009b. A generalized mixed effects model of abundance for mark-resight data when sampling is without replacement. In *Modeling Demographic Processes in Marked Populations*, D. L. Thomson, E. G. Cooch and M. J. Conroy, eds. Springer, New York, New York, USA, pp. 271-289.

McClintock, B. T., and White, G. C. 2012. From NOREMARK to MARK: software for estimating demographic parameters using mark-resight methodology. *Journal of Ornithology*, **152** (Suppl 2), S641-S650.

Minta, S., and Mangel, M. 1989. A simple population estimate based on simulation for capture-recapture and capture-resight data. *Ecology*, **70**, 1738-1751.

Neal, A. K., White, G. C., Gill, R. B., Reed, D. F., and Olterman, J. H. 1993. Evaluation of mark-resight model assumptions for estimating mountain sheep numbers. *Journal of Wildlife Management*, **57**, 436-450.

Otis, D. L., Burnham, K. P., White, G. C., and Anderson, D. R. 1978. Statistical inference from capture data on closed animal populations. *Wildlife Monographs*, **62**.

White, G. C. 1996. NOREMARK: population estimation from mark-resighting surveys. *Wildlife Society Bulletin*, **24**, 50-52.

White, G. C., and Shenk, T. M. 2001. Population estimation with radio-marked animals. In *Radio Tracking and Animal Populations*, J. Millspaugh and J. M. Marzluff, eds. Academic Press, pp. 329- 350.

CHAPTER 19

Young survival from marked adults

Paul Lukacs & Victoria Dreitz, *University of Montana*

The survival probability of juvenile animals is often of interest, but juvenile animals may be difficult to mark or marking may impact survival. In such cases, a count of the young animals with an attending adult may be the only data that can be obtained. If the adult is marked and highly detectable (such as with a radio transmitter), then survival of individual young can be estimated from the counts of young in a family group. This chapter describes the **MARK** implementation of the '**`Young Survival from Marked Adults`**' model developed in Lukacs *et al*. (2004).

The 'young survival model' is an extension of the Cormack-Jolly-Seber (CJS) model (which is discussed in detail in Chapters 1 → 7). The model contains two types of parameters, apparent survival (φ) and detection probability (p). Data for the CJS model consist of a string of ones and zeros indicating an individual animal is detected or not detected, whereas the data for the young survival model are a string of counts on young in a family group. Encounter histories are entered in the input file at the family group level rather than the individual young animal level. Despite this, it is important to note that the parameters refer to individual young animals not to the group in which they were detected. When there is only one young per family group, the young survival model reduces to the CJS model.

The young survival model is based on extending the CJS model to all possible combinations of events that could have occurred given the count of young observed. For example, in a clutch of three plover chicks that are known to be alive at hatching, consider a case where only two chicks are observed at the next sampling occasion. There are six events that could have occurred to produce that observation.

- First, two chicks survived and both were detected. This event occurs with probability $\varphi^2(1-\varphi)p^2$. The event can happen in $\binom{3}{2} = 3$ ways (because each of the 3 chicks starting the interval could have been the one that died), therefore the total probability of observing the event is $3\varphi^2(1-\varphi)p^2$.
- Second, all three chicks may have survived but only two were detected. This event occurs with probability $\varphi^3 p^2(1-p)$. The event also can happen in $\binom{3}{2} = 3$ ways (again because each of the 3 surviving chicks may have been the one not detected), therefore the total probability of observing that event is $3\varphi^3 p^2(1-p)$.

The probability of observing two chicks given three chicks were alive at the previous occasion is therefore $3\varphi^2(1-\varphi)p^2+3\varphi^3 p^2(1-p)$. Based on the combinatorics, it is easy to see that the number of possible events grows rapidly as the family group size increases. Therefore, precision decreases rapidly as the number of events increases.

© Cooch & White (2014)

19.1. Assumptions

The young survival model assumes that the number of young at the start of observation is known (if, for example, the number of eggs hatched out of a nest is counted exactly). If the initial size of the family group is not known and the entire family group is not observed at least once for each group in the data set, parameter estimates will be biased. Family groups can vary in size, but the initial size of each group must be known, and is assumed in **MARK** to be the maximum count observed (even if not the first count). **MARK** will function properly if the size of the family group is observed at least once during the study, but to assure parameter estimates are unbiased studies should be designed to know the family group size at the beginning of observation.

The young survival model also assumes that there is no brood mixing. Therefore, the number of young can never exceed the number the brood began with and new individuals cannot join the family group. Within a family group, young are assumed to be exchangeable, meaning that each individual animal has the same survival and detection probability as its siblings on a given occasion.

The parameter φ in the young survival model is defined as an *apparent* survival probability (the probability of an individual being alive and available for detection at occasion i given the individual was alive at i-1) rather than true survival (the probability of being alive at occasion i given the individual was alive at i-1) just as it is in the CJS model. In cases where separation of young from the attending adult results in death of the young, φ will essentially equal true survival. Study design should be carefully considered to separate fledging from mortality. Design considerations for this aspect will be highly species-dependent.

19.2. Data

Data for the young survival model consist of counts of young detected per family group on each occasion. The **MARK** input file uses the two digit count format similar to that of the Poisson log-normal mark-resight models (Chapter 18). Given this format, 0 to 99 young could be counted per group (although precision decreases greatly with more than 5 or so young per family group). Consider a case where a nest has 3 eggs hatch and 3, 2, 3, 0, and 1 young are counted over 5 occasions. The encounter history for this brood would be

```
0302030001                1;
```

Note that all counts must be entered as two digits with a leading 0 for values from 0-9 (for example 3 would be coded as '03'). If there are occasions where the marked attending adult is not located, and therefore the young have no chance of being located, two dots ('..') can be used to represent that information. For example, if the adult is not detected (or not sampled) on occasion 3, then the encounter history would be

```
0302..0001                1;
```

In all cases, a frequency (count of number of family groups with this encounter history) per encounter history (1 in this example) and a semicolon are needed as in other **MARK** data types.

The young survival model can incorporate covariates for more flexible modeling. Family group-specific covariates can be included just as individual covariates are included after the frequency column in the input file as in other **MARK** data types. Given individual young are not handled nor individually identifiable in this data collection scheme, only covariates measured on the whole family group will

be available. Environmental covariates (covariates acting on all family groups) can be included in the design matrix (presented in detail in Chapter 6).

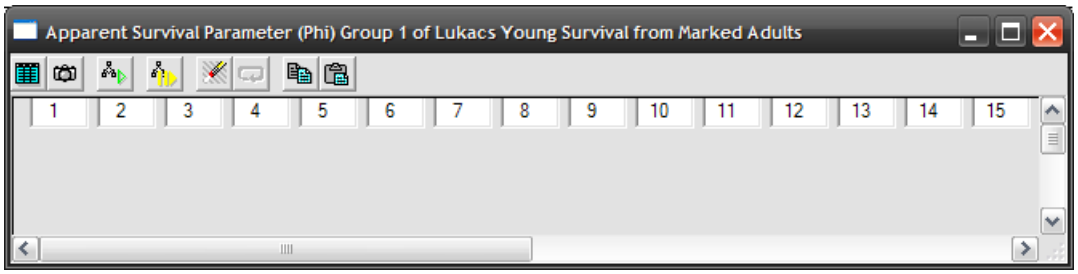
19.3. Model Implementation

To demonstrate the implementation of the young survival model in **MARK**, we use an example of mountain plover (*Charadrius montanus*) data simplified and modified from Dreitz (2009). The data consist of encounter histories of 31 occasions, one group and three covariates: year (three years coded in two covariates, with year 1 the '00' code)] and sex (0 = female, 1 = male) of the attending adult. The maximum number of chicks per family group is three in this case, because mountain plovers seldom or never have more than three eggs in a nest. Below are four encounter histories from the file `plover_mark.inp` showing the input file format.

```
0300..00......00....00..02..00....01..00..00..01..01..00..01..   1  0  1  0  ;
0301000203....03......01..01..010001..00....01..01..00..01..03   1  0  1  0  ;
01..00..00..01..01..00..0100..00..00..00......00..00..00..00..   1  0  1  1  ;
02000000....00....00......01....01..01..00..00.................  -1  0  1  0  ;
```

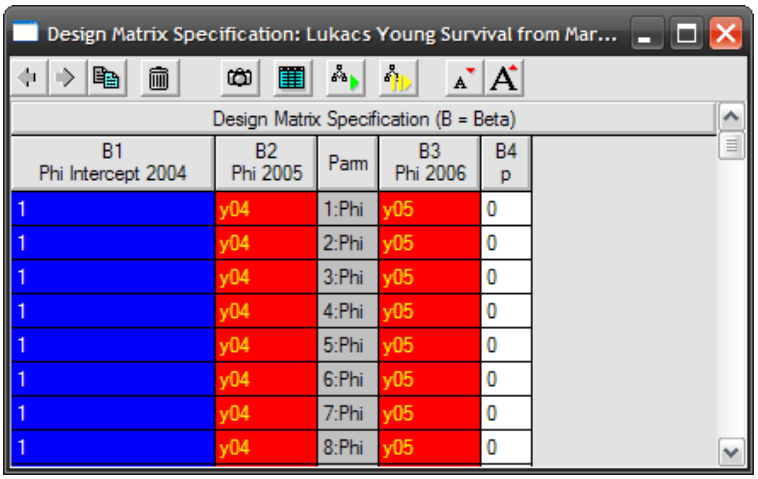
The fourth record has a '-1' in the frequency column indicating a family group that was censored at the last '00' because of a radio failure on the attending adult. However, because all the succeeding potential encounters are coded as '..', the '-1' value is equivalent to the usual '1'. Losses on capture in this data type are coded as '..' for the encounters after the loss, rather than as negative values of the frequency.

Model implementation for the young survival model follows that of the CJS model directly. The primary difference between implementing the young survival model and the CJS model in **MARK** is in the PIM format. The CJS model has triangular PIMs with rows representing cohorts while the young survival model has a single row in each PIM.



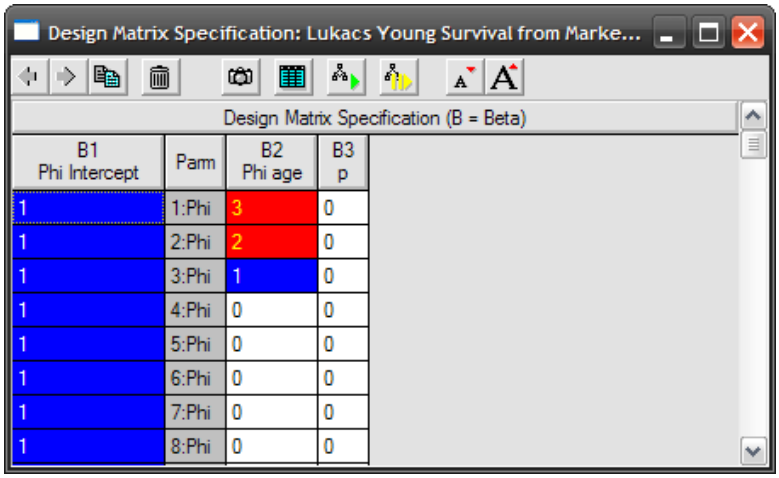
The single row PIM forces age-specific models to be built in the design matrix rather than the PIM, but otherwise does not limit the capability of **MARK**. On the whole, the single row PIM makes more sense than a triangular PIM for the young survival model because data generally consist of multiple occasions within a single year. Multiple years of data are better input as multiple attribute groups or as individual covariates indicating the year.

A year-specific survival model can be built in the design matrix with covariates as we demonstrate here (top of the next page) with the plover data. In this case `y05` and `y06` are binary covariates indicating data that occur in each year, when both covariates are 0 the data are from 2004.



B1 Phi Intercept 2004	B2 Phi 2005	Parm	B3 Phi 2006	B4 p
1	y04	1:Phi	y05	0
1	y04	2:Phi	y05	0
1	y04	3:Phi	y05	0
1	y04	4:Phi	y05	0
1	y04	5:Phi	y05	0
1	y04	6:Phi	y05	0
1	y04	7:Phi	y05	0
1	y04	8:Phi	y05	0

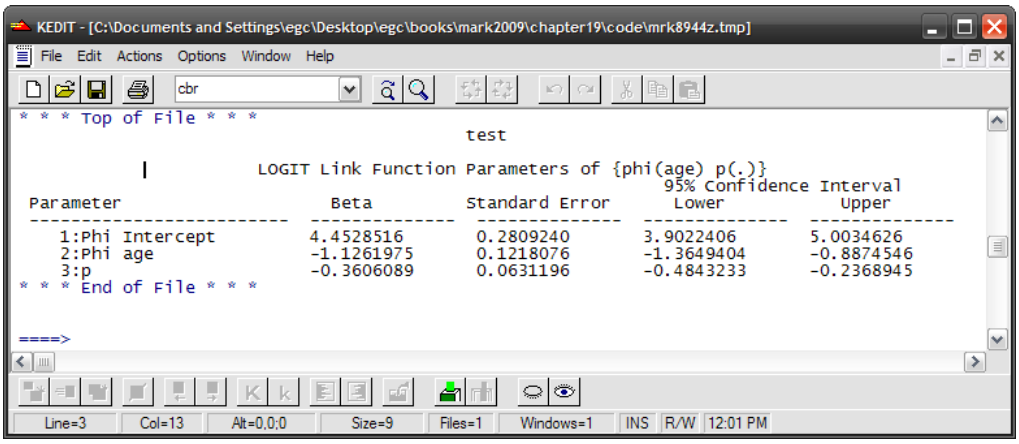
Many young animals have a lower survival probability immediately after they are born, but survival increases quickly thereafter. One way to model this is to place a trend on the first few occasions after birth. In our plover example, all family groups are started on the first occasion, so the implementation of the trend is simple.



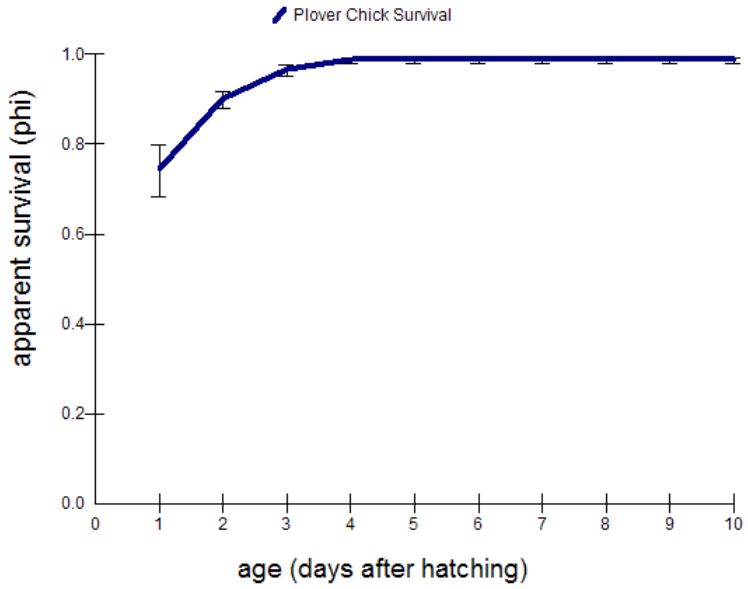
B1 Phi Intercept	Parm	B2 Phi age	B3 p
1	1:Phi	3	0
1	2:Phi	2	0
1	3:Phi	1	0
1	4:Phi	0	0
1	5:Phi	0	0
1	6:Phi	0	0
1	7:Phi	0	0
1	8:Phi	0	0

If family groups enter the data set on different occasions then a covariate would be needed to indicate the occasion of first observation. Likewise, occasions before the family group is entered into the analysis have to be coded with the '..' notation, because the '00' notation would suggest that the family group was sampled, and no young counted.

The age-specific model estimates suggest that survival is substantially lower immediately after hatching, but increases with age. The β parameter is strongly negative for the slope on the age parameter.



The real parameters for the $\{\varphi_{age} p_{.}\}$ model show the same effect on the probability scale.



19.4. Parameters and Sample Size

Parameter identifiability in the young survival model is the same as that in the CJS model. The last p and φ in a time-specific model are confounded. The use of fully time-specific models will be rare with this data type because occasions will often refer to days or other short time intervals.

Effective sample size for the young survival model is based on the number of "family group releases" (counts >0 and not '..') in the encounter histories. This matches the implementation of the CJS model in **MARK**, but differs from the way that effective sample size was computed in Lukacs *et al.* (2004).

Lukacs *et al*. (2004) used the number of encounter histories. The appropriate value for effective sample size in mark-recapture models remains an unresolved issue and more research on the topic is warranted. **MARK** allows the user to specify the value of effective sample size if desired by selecting the option under the '**Adjustments | Effective Sample Size**' menu.

19.5. Relationship with CJS and Multi-state Models

The young survival model has its roots in the multi-state and CJS models. When all family groups have only a single young, the 'CJS' and 'young survival' models are equivalent. This can be demonstrated with the cell probability for a '0101' encounter in the young survival model assuming a group size of 1 and the cell probability of a '11' encounter history for a CJS model.

$$\begin{aligned}\Pr[0101_{YS}] &= \begin{bmatrix}1 & 0\end{bmatrix}\begin{bmatrix}\varphi & (1-\varphi)\\ 0 & 1\end{bmatrix}\begin{bmatrix}p & 0\\ 0 & 0\end{bmatrix}\begin{bmatrix}1\\ 1\end{bmatrix}\\ &= \varphi p\\ &= \Pr[11_{CJS}]\end{aligned}$$

The relationship between the multi-state model and the young survival model is a bit less obvious. To demonstrate it, let us consider a case of two young per family group. First, we must define the states in terms of the number of young alive at any given occasion. Our three states are: *A*: 2 young live, *B*: 1 young live, *C*: 0 young live. Note that the states are defined by the true number of young alive not the observed number of young. When state definitions are based on whether or not an animal is alive, survival is modeled with the transition matrix and the multistate model survival probability is set to 1. Survival transitions can only proceed in one direction; therefore the lower left side of the matrix has all zero elements. The transition matrix takes the following form:

$$\Psi = \begin{bmatrix}\psi^{AA} & \psi^{AB} & \psi^{AC}\\ 0 & \psi^{BB} & \psi^{BC}\\ 0 & 0 & \psi^{CC}\end{bmatrix} = \begin{bmatrix}\varphi^2 & 2\varphi(1-\varphi) & (1-\varphi)^2\\ 0 & \varphi & (1-\varphi)\\ 0 & 0 & 1\end{bmatrix}$$

The capture probability matrix is more complicated. The states are defined by the true number of live young, but the observations are of young counted (some may be missed). Therefore, the only observation with a known state is that of a 2. There is state uncertainty in observations of 1 and 0. The detection probability matrix must account for the state uncertainty. For an observation of 1 young, the matrix is:

$$D(p) = \begin{bmatrix}2p(1-p) & 0 & 0\\ 0 & p & 0\\ 0 & 0 & 0\end{bmatrix}$$

This matrix handles the two possible states that an observation of 1 young could imply: either one young alive and detected, or two young alive with one detected and one not detected. A standard multi-state model would simply have a *p* for the state observed or (1-*p*) for all states if the animal was missed.

19.6. Summary

The young survival from marked adults data type in **MARK** allows apparent survival of young animals in a family group with a marked attending adult to be estimated. The model only requires counts of young in the family group rather than individually marked young. When designing a study, it is important to consider trade-offs of counting young versus marking young. As long as the marking process does not affect the survival of the young, estimates of survival from marked young will be more precise than that from an equal number of unmarked young. It may be possible to count more family groups of unmarked young than can be marked; in that case weighing the tradeoff requires a more careful examination of the benefits of marks versus a larger sample size.

References

Dreitz, V. J. (2009) Parental behaviour of a precocial species: implications for juvenile survival. *Journal of Applied Ecology*, **46**, 870-878.

Lukacs, P. M., V. J. Dreitz, F. L. Knopf, and K. P. Burnham. (2004) Estimating survival probabilities of unmarked dependent young when detection is imperfect. *Condor*, **106**, 927-932.

CHAPTER 20

Density estimation...

Jake Ivan, *Colorado Parks and Wildlife*

Abundance is commonly sought after as a state variable for the study of populations. However, density (number of animals per unit area) can be a more meaningful metric because it casts the state of a population into a common currency. For example, using closed capture models from Chapter 14, we may estimate 500 animals at site A and 200 animals at site B. Thus one conclusion we may reach is that habitat management at site A has positively impacted the population there compared to site B. However, if we know site A is 250 hectares and B is 100 hectares, then we realize that each has 2.0 animals/hectare. That is, on a relative scale, the different management scheme at A had no effect compared to site B. Conversely, we may estimate abundance at 2 sites to be similar and conclude management actions, or habitat types, or harvest regulations, etc. are having a similar impact, but if the sites are different sizes, then the impacts are actually quite different on a relative scale and our conclusion is erroneous. Thus, while abundance can be a useful metric, estimating density can be helpful as well.

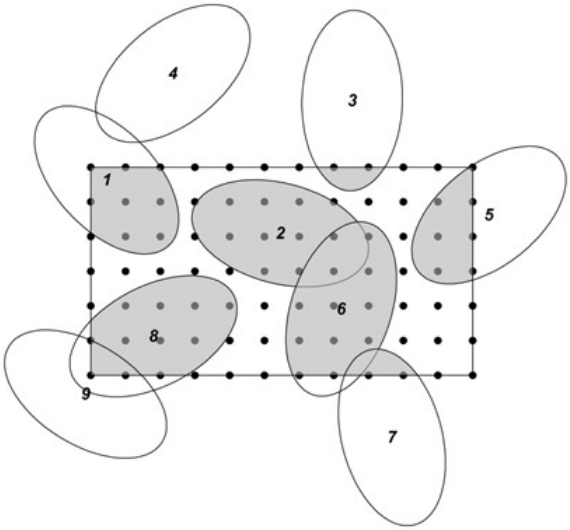
Chapter 14 covered in depth how one might use mark-recapture techniques to obtain an estimate of abundance from closed population samples. And in this age of remote imagery, GPS, and GIS, it is relatively easy to delineate a study site and compute its area. Therefore, it should be easy to compute the number of animals per unit area, right? Simply obtain an estimate of abundance using any one (or several) of the multitude of models covered in Chapter 14, then divide that estimate by the area of the study site. Certainly you can proceed in this straightforward manner, and sometimes people do. However, your estimate will likely be biased. Why? Recall the assumptions of closed capture models. Not only do we need the marks to be individually identifiable, readable, and permanent (at least over the duration of the sampling), in order for our inference to be what we intend it to be, we need the population we're sampling to be closed both demographically and geographically.

Demographic closure can often be met by carefully considering the natural history of the species of interest, then timing the sampling to correspond to periods when births and immigration/emigration (e.g., dispersal) are unlikely. Sampling over a relatively short duration can give you some assurance that deaths during the sampling period should be relatively unlikely as well. However, achieving geographic closure is much more difficult. In fact, given that wildlife are mobile, we likely violate the geographic closure assumption in most applications. If you happen to be sampling a terrestrial species on a small island, you're probably fine. Or, inversely, if you're sampling fish in a pond, you probably meet the assumption in that case as well. You may even be able to reasonably assume closure in other select situations, such as sampling forest dwelling rodents in small woodland patches that are surrounded by a matrix of agriculture fields.

However, in the vast majority of situations, we sample wildlife populations in relatively continuous

© Cooch & White (2014)

habitat such that some animal home ranges (simplistically depicted below as ovals) are completely within the boundaries of our study site (i.e., the home range for animal 2 below falls completely within the study site [rectangle]), but many others only partially overlap it (animals 1, 3, 5, 6, 7, 8, 9). If this is the case, many individuals, especially those near the edges of the site, will move on and off of the area of interest *while mark-recapture sampling is ongoing*.



For example, during a traditional live-trapping study of small mammals in the fictitious situation depicted above, animal 5 may be on the site and available for capture during the 1st day of trapping (occasion 1), then off of the study area for day 2, back on for days 3 and 4, then off again on day 5. Assume for a moment that capture probability is perfect ($p = 1.0$). That is, whenever this animal is on the study area, it gets captured. Under this scenario, the encounter history for animal 5 would be '`10110`'. A closed capture model, however, assumes that if animal i was captured on the study area on any occasion, it was available on the study area for all occasions (i.e., the study area is closed - once an animal is on it, it can't move off). Given this assumption, the '`10110`' capture history would indicate that p is closer to 0.6 (this animal was available on 5 occasions and was detected on 3 of them). Thus, when we lack geographic closure, closed capture models will underestimate capture probability. In fact, the estimate returned from closed capture models is actually a product. It's the product of the probability that animal i is available for capture (a) and the probability that animal i is detected given it was available (p) such that $p_{\text{estimated}} = p \times a$. When we assume closure, we assume $a = 1.0$; when the assumption is violated $a < 1.0$ and $p_{\text{estimated}} < p$.

Now, recall from the beginning of Chapter 14 that the basic, heuristic estimator for abundance for a single occasion is:

$$\hat{N} = \frac{n}{\hat{p}}$$

where n is the number of unique animals observed (the count statistic) and $\hat{p}$ is the estimated probability that an individual is detected. It stands to reason that if $\hat{p}$ is underestimated when we lack closure, then our estimate of $\hat{N}$ will be inflated. Thus, when we fit a closed capture model to data from a design in which the study site is not closed, we do not obtain an estimate of the number of animals on the study site, which is what we want. Our estimate is larger than that. In fact, what we estimate is the number

of animals that could have used the study site during the course of sampling. We term this the super population (Schwarz and Arnason 1996, Kendall *et al.* 1997). Is this a problem? Maybe, maybe not. In some cases you may elect to simply re-define what it is you're estimating and report that number.

You can imagine, however, that if your goal is to estimate density, lack of geographic closure is problematic. We have an estimate of abundance, but we do not know area to which that estimate applies. Intuitively, the area used by the super population during the course of sampling was larger than the study area itself, but how much larger? In other words, in the expression for density

$$\hat{D} = \frac{\hat{N}}{A}$$

what value do we use for A? The choice is not clear, yet the choice can have a big effect on your estimate of density. Problem? Problem. What to do? Well, there are options. In fact, scientists have been proposing solutions to this very problem since the inception of the field of wildlife ecology in the 1930s. These include using home range estimates to take a stab at the effective area sampled (Dice 1938), exploiting differences in capture rates (or abundance estimates) between inner and outer detectors at the site (e.g., Maclulich 1951, Hansson 1969, Otis *et al.* 1978), using assessment lines to determine the reach of the initial sampling effort (e.g., Smith *et al.* 1971, Van Horne 1982), or computing the distances moved by individuals between detection events (Otis *et al.* 1978, Wilson and Anderson 1985). However, most of these approaches have fallen out of favor over time due to logistical issues, unrealistic data requirements, or *ad hoc* rather than theoretical foundations. A notable exception is the method based on mean maximum distance moved between trapping events (Otis *et al.* 1978, Wilson and Anderson 1985b), which has received criticism as an ad hoc approach (Williams *et al*. 2002, Royle and Dorazio 2008), and has shortcomings (Parmenter *et al.* 2003), but is still fairly popular (e.g., Converse *et al*. 2006, Zahratka and Shenk 2008).

Spatially explicit capture-recapture (SECR; Efford 2004, Borchers and Efford 2008, Royle and Dorazio 2008, Efford *et al.* 2009, Royle *et al.* 2009) and trapping webs (Anderson *et al*. 1983, Link and Barker 1994) are two contemporary density estimation approaches that circumvent the difficulties of estimating the area used by the super population by estimating density directly. These approaches generally have a stronger theoretical background behind them than the approaches mentioned above, but like any model, they have their own sets of assumptions and thus advantages and disadvantages. See (Parmenter *et al*. 2003) and (Ivan *et al*. 2013) for discussions of pros and cons, advantages, and limitations.

Here we focus on the solution provided to you in Program **MARK**. It is called the "**`Density with Telemetry`**" data type. From the name you can see right away that this approach requires auxiliary information (telemetry locations). Thus, as with the Barker model and Burnham's joint model, we are combining mark-recapture information with outside information to help solve a problem. The basic approach is exactly opposite of that taken during earlier work described above. That is, rather than estimating N, then trying to figure out the area to which the estimate applies, we first define the study site of interest, and thus fix its area. We then attempt to estimate the total number of whole and partial animals within its boundaries. In the case of our simple example above, this amounts to using telemetry to estimate the shaded areas for each of the 9 animals that were part of the mark-recapture data set. We then sum these proportions (i.e., for animals $1 \rightarrow 9$, we might estimate there are a total of $[0.47 + 1.00 + 0.08 + 0.00 + 0.37 + 0.93 + 0.09 + 0.82 + 0.07] = 3.83$ animals on the study site) and divide by the area of the study site to obtain an unbiased estimate of density.

The basic setup in the field is as follows. Assume that our sampling scenario is qualitatively similar to the simplistic example above. That is, despite our effort to nicely demarcate the study site of interest in a GIS, we know wildlife will go about their daily business largely ignoring our boundaries. Also assume we sample animals at the study site during a time of year in which we are not concerned about

dispersal or births and we sample over a short enough time span that we're not so much concerned with deaths either (i.e., the population is demographically closed). During a sampling session, we capture, mark, and release individuals on multiple occasions, let's say 5. Let's also suppose we fit these captured animals with a telemetry device (e.g., radio tag). We start collecting locations on telemetered individuals after we're done with the mark-recapture sampling and we sample locations from these animals over a relatively short period of time, say 2 weeks post-trapping. For each independent location, we note whether it is within the study site, or outside the study site.

We know we can apply traditional mark-recapture estimators to the mark-recapture data collected during the sampling session to estimate the super population (Schwarz and Arnason 1996, Kendall *et al.* 1997) of animals that used the site during sampling. Our goal is to use the telemetry data to estimate the portion of the super population that occurred within the boundaries of the site to produce an estimate of density corrected for the lack of geographic closure. Mathematically, we begin with the Huggins (1989;1991) closed capture estimator for abundance:

$$\hat{N}_{sp} = \sum_{i=1}^{M_{t+1}} \frac{1}{\hat{p}_i^*}$$

where $\hat{N}_{sp}$ is the abundance estimate that represents the *super population* of animals that could have used the site during the trapping session, $\hat{p}_i^*$ is the estimated probability that animal i is captured one or more times during the sampling session (i.e., if $\hat{p}_i$ is an estimate of the probability animal that i is detected on any given occasion, then $\hat{p}_i^* = 1 - (1 - \hat{p}_i)^t$, where t is the number of occasions), and M_{t+1} is the total number of animals detected. This should look very familiar if you've read Chapter 14, with the exception that we're explicitly noting that in most cases, $\hat{N}$ will be an estimate of the super population rather than an estimate of the number of animals within the study site.

Next let's make a substitution in the numerator of the Huggins estimator:

$$\hat{N}_{ss} = \sum_{i=1}^{M_{t+1}} \left(\frac{\hat{\hat{p}}_i}{\hat{p}_i^*} \right)$$

where $\hat{\hat{p}}$ is the estimated proportion of time (i.e., proportion of telemetry locations) animal i spent on the study site. Those individuals that are always located on the study site contribute fully to the estimate and are assigned $\hat{\hat{p}}_i = 1$. That is, they are treated no differently than they are in the normal Huggins implementation. Most other individuals receive a fractional $\hat{\hat{p}}_i$ because they spend some fraction of their time on the site. Depending on the circumstances, some (many?) animals may be like animal 4 above. That is, they may be attracted to our detectors and become part of the mark-recapture data set, especially if we use bait or lures to improve our detection probability, but in reality their normal home range doesn't occur within the study site. These animals will be assigned $\hat{\hat{p}}_i = 0$ and do not contribute to the estimate.

So, functionally what we have here is an estimator that adjusts our count of animals detected upward (i.e., divide by $\hat{p}_i^*$) to accommodate imperfect detection, then ratchets that correction back down (i.e., multiply by $\hat{\hat{p}}_i$) to include only those animals within the study area. $\hat{N}_{ss}$, then, is the estimated number of animals within the study site, which is what we've always wanted our abundance estimate to be.

At this point all we need to do is divide $\hat{N}_{ss}$ by the area of the study site (e.g., area of the minimum convex polygon encompassing all detectors. It is this polygon that is used to determine whether animals are 'in' or 'out', which informs derivation of $\tilde{p}$) in order to obtain an estimate of density:

$$\hat{D} = \frac{\sum_{i=1}^{M_{t+1}} \left(\frac{\hat{\tilde{p}}_i}{\hat{p}_i^*} \right)}{A}$$

where $\hat{D}$ is estimated density (number of animals per unit area), A is the area of the study site, and $\hat{\tilde{p}}_i$, $\hat{p}_i^*$, and M_{t+1} are as defined previously. **MARK** takes care of the variance calculation for you. Note that you do not necessarily have to put a telemetry device on each of the animals you sample via mark-recapture. If it is not possible to telemeter all individuals, we can use logistic regression to build a relationship between $\hat{\tilde{p}}_i$ and a suite of covariates for animals that do have a telemetry device. We can then use this logistic model to estimate $\hat{\tilde{p}}_i$ for those individuals that were not telemetered. See Ivan *et al*. (2013) for a summary of estimator performance for different levels of telemetry effort.

20.1. Likelihood

The likelihood for this estimator is simply a combination of likelihoods you have seen before. Chapter 14 has extensive detail regarding the structure of the likelihood for various forms of the Huggins estimator (i.e., the likelihood with respect to parameters p, c, and potentially π). Chapter 16 describes the likelihood for known fate survival models, which is a simple binomial - the same model used for logistic regression. In Chapter 16, the parameter we were trying to estimate was survival (S), success was defined as surviving an interval, failure was dying, and N, the number of trials (not to be confused with abundance, $\hat{N}$), was the number of animals alive to start each interval. In the case of density estimation, $\tilde{p}$ replaces S as the parameter of interest, a binomial "success" is locating an animal on the study site (i.e., the polygon), a "failure" is locating it off of the study site, and the number of trials, N, is the number of total locations collected (on an individual basis - the N's need not be the same for each individual). The two likelihoods can be written out separately, then multiplied together to form one big likelihood, which is then maximized.

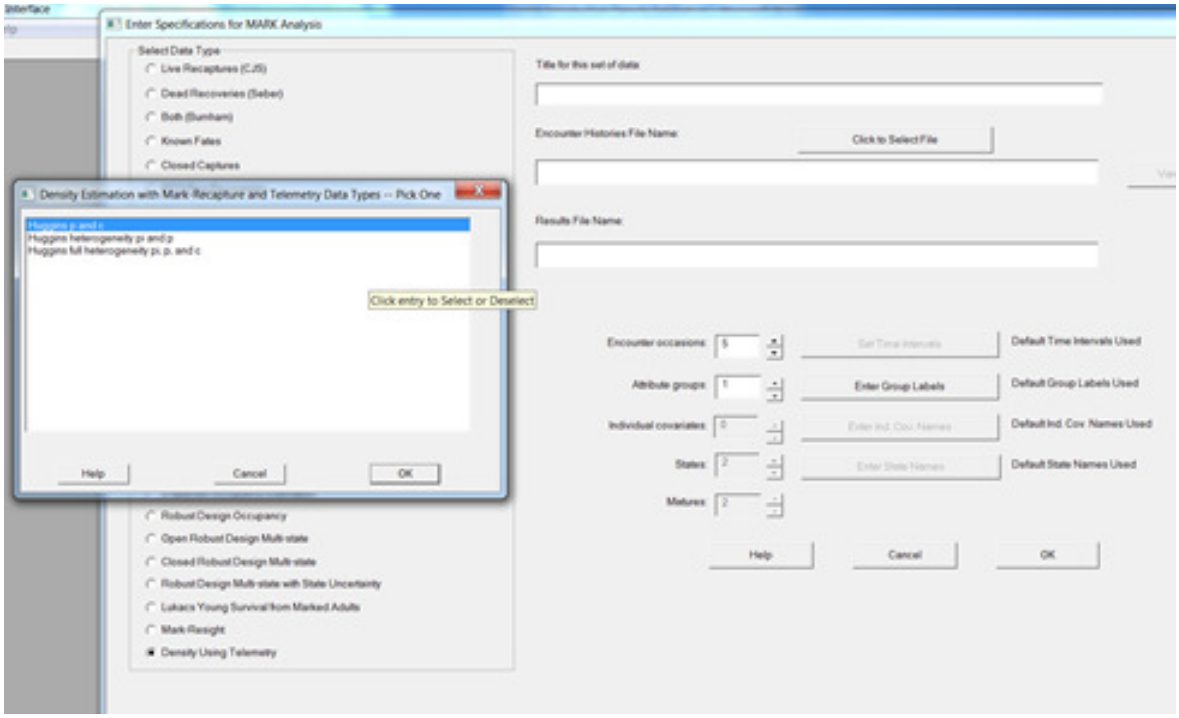
20.2. Implementation in MARK

With this conceptual and mathematical background in mind, let's look at the mechanics of how to implement the model in **MARK**. First, the input file. As you may have guessed, it looks generally like a closed capture input file with a slight modification to accommodate the telemetry data. A piece of an example input file appears below. As per the usual closed capture format, each line begins with an optional comment (here an animal ID #), followed by the encounter history (in this case, 5 occasions). Next are 2 new columns where we input the telemetry data. The first of these new columns is the number of times animal i was located on the study site; the second column is the total number of telemetry locations obtained on animal i. So, in the example, animal 20 was located on the study site 100% of the time (10 locations on site out of 10 total locations collected), animal 24 was located on the site 50% of the time (5/10), and animal 26 was never located within the bounds of the study site after having been captured and tagged there (0/10). Note also that animals 27 and 28 were part of the mark-recapture sample, but they were not fitted with telemetry devices. For those animals, we enter "." for each of the 2 telemetry columns as these animals provide no information for that part of the model. The remaining columns are the same as they would be for the usual closed capture input file. In this case,

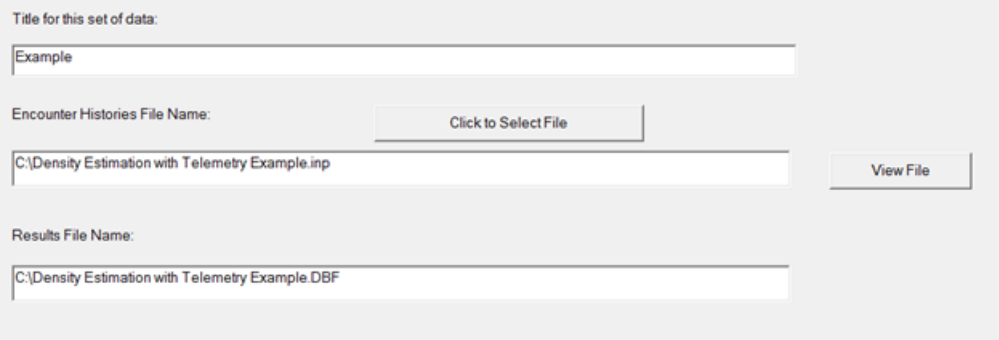
after the columns for the telemetry data, we have 2 groups (animals 20-24 belong to group 1, animals 25-28 belong to group 2), then a single covariate, which we will discuss later.

```
/*20*/        10010  10  10  1   0   20;
/*21*/        10000  1   10  1   0   0;
/*22*/        00100  8   10  1   0   20;
/*23*/        00001  10  10  1   0   20;
/*24*/        00010  5   10  1   0   0;
/*25*/        00010  5   10  0   1   20;
/*26*/        00010  0   10  0   1   0;
/*27*/        00110  .   .   0   1   25;
/*28*/        00001  .   .   0   1   0;
```

To load the input file, open **MARK**, choose "**`File | New`**" and select the "**`Density with Telemetry`**" data type near the bottom of the screen. A window appears in which you will have to select one of 3 possible parameterizations for the capture probability parameter(s). Note that these options are the same as those available to you under the traditional Huggins closed capture data type (Chapter 14). For the sake of this example, let's keep it relatively simple and choose "**`Huggins p and c`**".

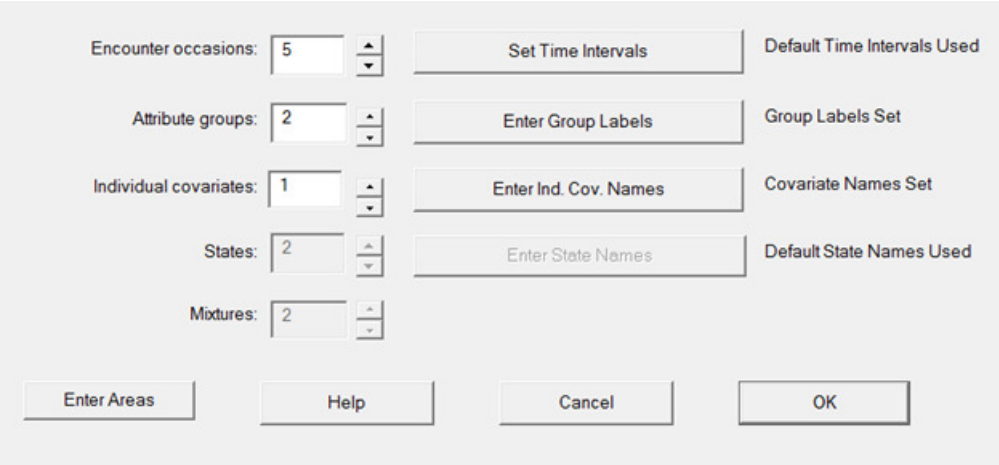


Once we've selected the parameterization we'd like to use to model capture probability, we need select an input file and fill in the rest of the Specifications Window as per usual. Let's title this data "`Example`", and choose the "`Density Estimation With Telemetry.inp`" input file provided with **MARK**.



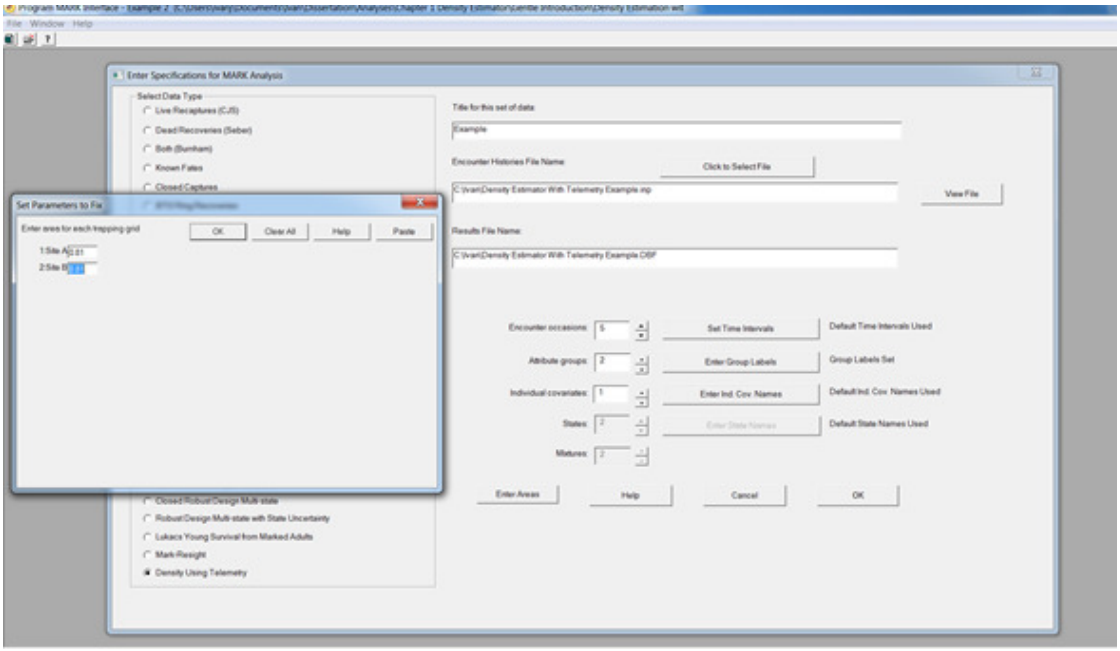
This is simulated data intended to mimic a study of small mammals, such as deer mice, sampled in 2 sites (habitat types), A and B. Each habitat type was sampled with a (10×100) live-trapping grid (10m trap spacing). There are 5 occasions. In addition to marking each mouse with an individually identifiable ear tag, 50% of the individuals captured were fitted with a small VHF transmitter. These radio-tagged individuals were located once during the day and once at night for 5 days immediately after mark-recapture sampling (n = 10 locations total per animal) and each location was recorded as "in" or "out" of the study site. The single covariate we've recorded is the distance to the edge (DTE) of the of the site from the mean trap location of each individual (i.e., compute the mean trap location for each individual captured ≥ 1 time, then compute the minimum distance from this mean location to the edge of the site). This is a crude metric for characterizing whether an individual's home range is near the edge or in the interior of the site, which can be helpful in modeling both the p and $\tilde{p}$ parameters as we'll see below.

Given this information, fill in the remainder of the "**`Specifications Window`**" as follows:



Label the 2 groups "`Site A`" and "`Site B`"; label the Individual Covariate "`DTE`". Note that with this data types you are required to enter Group Labels. You cannot accept the defaults as you can with other data types. Also, note that there is a new "**`Enter Areas`**" button at the bottom of this window. You must click this button to specify the area of the study site for each group before you can proceed. The area of the site is critical to the final computation, and it defaults to 1. It is also allowed to vary with each group. Forcing you to double check the labels and corresponding areas before you proceed is **MARK**'s way of imparting some quality control on your project!

You should enter the area for each group in whatever units you would like the density estimate expressed. In our example, each site (group) was sampled with a (10×10) grid with 10m trap spacing and we want the answer to be in animals per hectare so we enter 0.81 for Sites A and B (i.e., let's define the study site as the minimum convex polygon around the traps so the site is $(90\text{m} \times 90\text{m})$. See sidebar below for more discussion.). If we wanted the answer to be animals/m^2, we would enter 8,100 for each group; for animals per kilometer, we would enter 0.0081 for each group, etc. If Site B was smaller, say 0.75 ha, we would enter 0.81 for the area of Site A and 0.75 for the area of Site B.

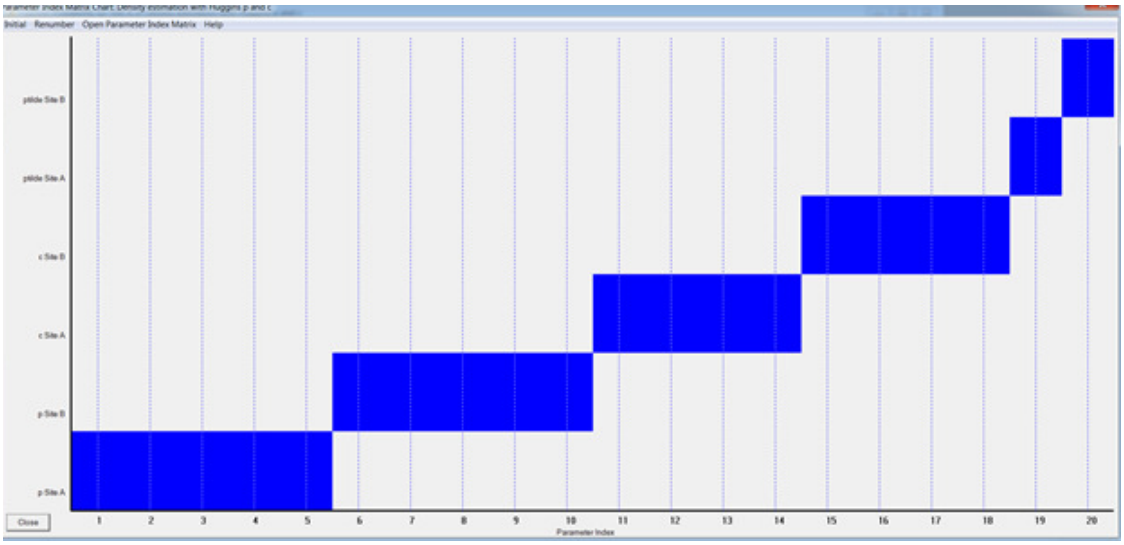


begin sidebar

Note that the definition of the site is somewhat arbitrary. However, any workable definition should ensure that animals within the site have a reasonable chance of detection. We suggest that a "reasonable chance of detection" can be attained if the site is defined such that detectors are distributed at an adequate and roughly consistent rate (i.e., 4 detectors per home range; Otis *et al.* 1978, p. 76) within it so that there are no gaps in sampling effort. We prefer to define the study site as the minimum convex polygon (MCP) encompassing the detectors. Such a definition seems natural and trap density inside this polygon would meet our criteria of being relatively high and consistent throughout. Alternatively, one could define the study site as the MCP plus ½ trap-width, or a full trap-width, and still claim that trap density is relatively high and consistent within the site. However if we defined the site as the MCP plus 2-3 trap-widths or if detectors were distributed at a rate of 2 per home range over part of the site, but 6 per home range over other parts, those definitions would likely result in unequal sampling effort across the site and may lead to bias in the estimator. Note that because we incorporate the proportion of time each individual spends on the site (i.e., telemetry data), the estimator will be appropriate regardless of how the site is defined, as long as it is defined following the guidance we provide here.

end sidebar

Once you've entered the area for each group, **MARK** will allow you to click '**OK**' and proceed. In the main **MARK** interface, open the PIM chart to view the parameters in the model. You will see the familiar p's (5 for each group) and c's (4 for each group) from the Huggins portion of the model along with the new parameter we're going to model using the telemetry data, $\tilde{p}$ (1 for each group).



Because the Huggins model is embedded within this model for density, all of the Huggins options from Chapter 14 are available to you for modeling p and c. That is, you can specify that capture probability is constant across animals and occasions (M_0), or that it varies by occasion (M_t), or that it differs depending on whether an animal has been captured previously (M_b), or that it differs generally among individuals (M_h), or that it varies in relation to one or more individual covariates. Of course, you can specify any combination of these basic model types as well. All of the nuances covered in Chapter 14 regarding model construction for closed abundance estimators(e.g., constraining the last p) apply here as well.

You may initially be tempted to structure similar models for $\tilde{p}$, but think before you start filling in your design matrix or PIM chart. For $\tilde{p}$ (proportion of time, or proportion of locations on the study site), there is no notion of a behavioral effect, nor time effects (we generally sample through time to estimate $\tilde{p}$, but $\tilde{p}$ is not connected to the mark-recapture sampling occasions in any way). Heterogeneity in $\tilde{p}$ is possible, and even likely, given that some animals will be located on the edge of the study site whereas others have home ranges near the interior. However, we cannot model general heterogeneity using a mixture model (π) like we do with capture probability. Instead, we're limited to the use of covariates, such as DTE. Thus, outside of covariates or groups, the structure we can apply to $\tilde{p}$ is limited.

Leave the PIMs in this general form, close the PIM chart, and open a '**reduced**' design matrix with 2 columns. Let's begin by building the simplest model possible: $p(.)\tilde{p}(.)$ (which we write in ASCII as `p(.)p~(.)`). For this first model, we're specifying a constant detection probability (M_0) and we're assigning each animal the average proportion of time on site across all animals and groups - akin to the basic form of this estimator first introduced by White and Shenk (2001). This may not be the most biologically realistic model we can think of, but it's a basis for comparison and may be the most supportable structure available if you have a sparse data set. The design matrix for this model is shown at the top of the next page:

B1: p int	Parm	B2 p~ int
1	1:p	0
1	2:p	0
1	3:p	0
1	4:p	0
1	5:p	0
1	6:p	0
1	7:p	0
1	8:p	0
1	9:p	0
1	10:p	0
1	11:c	0
1	12:c	0
1	13:c	0
1	14:c	0
1	15:c	0
1	16:c	0
1	17:c	0
1	18:c	0
0	19:ptilde	1
0	20:ptilde	1

Run this model then examine the real parameters. We see that capture probability was estimated to be 0.23 and $\tilde{p}$ was 0.66. So, on average, each animal was located on their respective study site about 2/3 of the time according to this model. The density estimate for the 2 sites is a derived parameter. If we examine the derived estimates we notice that there appears to be some difference in density between the 2 study sites (21.63 animal/ha in Site A, 14.80 animals/ha in Site B). Note, however, that because density is a derived parameter, we cannot specifically test for differences in density between sites by specifying a model structure that allows for differences and comparing it to one that does not. We can specify differences in p or $\tilde{p}$, but not $\hat{D}$. Instead we're left to make inference by other means. That is, we have to examine the point estimates, SEs, and confidence intervals, then make a judgment. In this case, the point estimates appear to be fairly different, but there is substantial overlap in the 95% confidence intervals, so maybe we conclude there is some evidence that density at Site A is higher than Site B, but the evidence is weak.

Example

Real Function Parameters of {p(.)p~(.)}

Parameter	Estimate	Standard Error	95% Confidence Interval Lower	Upper
1:p	0.2252994	0.0424134	0.1529914	0.3189148
2:p	0.2252994	0.0424134	0.1529914	0.3189148
3:p	0.2252994	0.0424134	0.1529914	0.3189148
4:p	0.2252994	0.0424134	0.1529914	0.3189148
5:p	0.2252994	0.0424134	0.1529914	0.3189148
6:p	0.2252994	0.0424134	0.1529914	0.3189148
7:p	0.2252994	0.0424134	0.1529914	0.3189148
8:p	0.2252994	0.0424134	0.1529914	0.3189148
9:p	0.2252994	0.0424134	0.1529914	0.3189148
10:p	0.2252994	0.0424134	0.1529914	0.3189148
11:c	0.2252994	0.0424134	0.1529914	0.3189148
12:c	0.2252994	0.0424134	0.1529914	0.3189148
13:c	0.2252994	0.0424134	0.1529914	0.3189148
14:c	0.2252994	0.0424134	0.1529914	0.3189148
15:c	0.2252994	0.0424134	0.1529914	0.3189148
16:c	0.2252994	0.0424134	0.1529914	0.3189148
17:c	0.2252994	0.0424134	0.1529914	0.3189148
18:c	0.2252994	0.0424134	0.1529914	0.3189148
19:ptilde	0.6647059	0.0362079	0.5904713	0.7316007
20:ptilde	0.6647059	0.0362079	0.5904713	0.7316007

Estimates of Derived Parameters
Density Estimates of {p(.)p~(.)}

Group	D-hat	Standard Error	95% Confidence Interval Lower	Upper
1	21.626592	4.7100393	14.112441	33.141644
2	14.797142	3.7073721	9.0554150	24.179501

We might hypothesize differences in p and/or $\tilde{p}$ between the sites if the habitat imparts differential home range sizes and/or differential availability of food resources that impact detection. Let's go ahead and build 2 models that reflect these hypotheses, that p or $\tilde{p}$ may differ by site. Recall from above, that Site A (group 1) is represented by parameters 1-5 (p), 11-14 (c), and 19 ($\tilde{p}$); Site B (group 2) is represented by parameters 6-10 (p), 15-18 (c), and 20 ($\tilde{p}$). Thus, a potential model that reflects the hypothesis that p is different between sites might look like the design matrix on the left, whereas the one reflecting differences in $\tilde{p}$ might look like the one on the right. Of course, the 2 structures could be combined as well.

B1: p int	Parm	B2: p site	B3 p~ int
1	1:p	0	0
1	2:p	0	0
1	3:p	0	0
1	4:p	0	0
1	5:p	0	0
1	6:p	1	0
1	7:p	1	0
1	8:p	1	0
1	9:p	1	0
1	10:p	1	0
1	11:c	0	0
1	12:c	0	0
1	13:c	0	0
1	14:c	0	0
1	15:c	1	0
1	16:c	1	0
1	17:c	1	0
1	18:c	1	0
0	19:ptilde	0	1
0	20:ptilde	0	1

B1: p int	Parm	B2: p~ int	B3: p~ site
1	1:p	0	0
1	2:p	0	0
1	3:p	0	0
1	4:p	0	0
1	5:p	0	0
1	6:p	0	0
1	7:p	0	0
1	8:p	0	0
1	9:p	0	0
1	10:p	0	0
1	11:c	0	0
1	12:c	0	0
1	13:c	0	0
1	14:c	0	0
1	15:c	0	0
1	16:c	0	0
1	17:c	0	0
1	18:c	0	0
0	19:ptilde	1	0
0	20:ptilde	1	1

In this case, the model that allows $\tilde{p}$ to vary between sites has more support than the one that allows p to vary between sites. If we examine the β estimates for the best fitting model thus far, we see that with Site A as the reference point (intercept), Site B tended to have lower $\tilde{p}$. That is, animals on that site tended to spend less time within the study boundaries compared to animals at the site represented by the intercept (Site A).

Example

LOGIT Link Function Parameters of {p(.)p~(site)}

Parameter	Beta	Standard Error	95% Confidence Interval Lower	Upper
1:p int	-1.2350464	0.2430015	-1.7113294	-0.7587635
2:p~ int	1.0459686	0.2279804	0.5991270	1.4928102
3:p~ site	-0.8163941	0.3314727	-1.4660807	-0.1667076

Finally, let's run a model that makes use of our DTE covariate. We might expect that if an animal is trapped, on average, near the center of the study site, it may have a higher probability of detection than

animals trapped near the edge because there are more traps in its home range compared to the edge animal. Similarly, we might also expect such an individual to be located on the study site more often than an animal whose home range is near the edge of the site. So, $p(DTE)\tilde{p}(DTE)$ (i.e., `p(DTE)p~(DTE)`) is a candidate model we may want to run, and we expect the DTE covariate to be positively related to both parameters. If we run this model, we see that our hypothesis was well supported. This model is just over 50 AIC_c units better than our previous best fitting model. Also, as expected, the relationship between DTE and both p, and $\tilde{p}$ was positive. Note that this covariate can be computed for most any closed capture or density estimation setup. Keep it in mind as it often proves useful.

Model	AICc	Delta AICc	AICc Weight	Model Likelihood	No. Par.	Deviance	-2Log(L)
{p(DTE)p~(DTE)}	351.0997	0.0000	1.00000	1.0000	4	342.8416	342.8416
{p(.)p~(site)}	401.1442	50.0445	0.00000	0.0000	3	394.9904	394.9904
{p(.)p~(.)}	405.2051	54.1054	0.00000	0.0000	2	401.1286	401.1286
{p(site)p~(.)}	405.5486	54.4489	0.00000	0.0000	3	399.3948	399.3948

Example

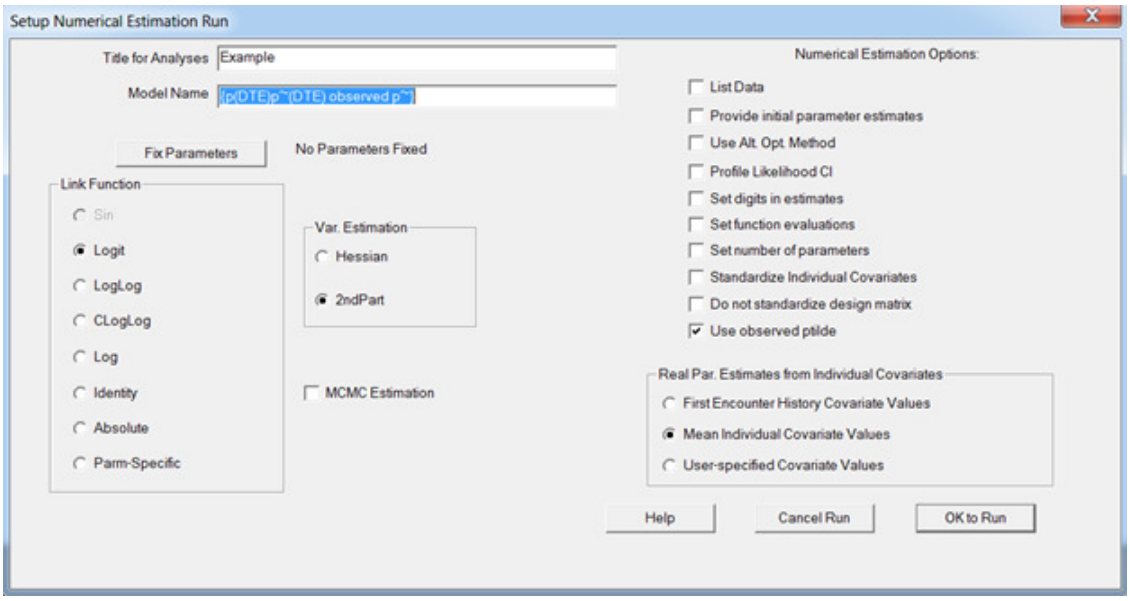
LOGIT Link Function Parameters of {p(DTE)p~(DTE)}

Parameter	Beta	Standard Error	95% Confidence Interval Lower	Upper
1:p int	-2.3480221	0.5668408	-3.4590300	-1.2370142
2:p DTE	0.0752421	0.0295541	0.0173160	0.1331682
3:p~ int	-0.6218991	0.2554951	-1.1226696	-0.1211287
4:p~ DTE	0.1450665	0.0241161	0.0977990	0.1923340

20.2.1. Estimate proportion on site or use the data?

To this point we've been accepting the default run options in **MARK**, such that $\tilde{p}$ is estimated using logistic regression. This means that we've been specifying a model that relates the proportion of time on site to an intercept, or an intercept plus group indicators, or an intercept plus covariates, etc. MARK then uses this relationship to estimate $\tilde{p}$ for the animals that were sampled via mark-recapture but not telemetry (i.e., those individuals that had "`. .`" for the 2 telemetry columns in the input file) as *well* as those that were sampled via telemetry. In other words, the resulting logistic model gets applied to each of the i animals to derive an estimate of $\tilde{p}$ which is then summed over all individuals. You may be wondering why you wouldn't just use the telemetry data itself to estimate $\tilde{p}$ for those animals that were telemetered, then use logistic model for only those animals that weren't telemetered.

Actually, you have that option. Retrieve `p(DTE)p~(DTE)` in the browser and click the run button. Notice that in the run window (shown on the next page) you have an extra option for this data type. You can check a box to "**`Use observed ptilde`**". Let's check the box this time and re-run the model. Add the label "**`observed ptilde`**" to the name so we can keep it straight.



Notice (shown below) that this model has the exact same number of parameters, same log likelihood, and same AIC_c as the original `p(DTE)p~(DTE)` model. This is because **MARK** still has to fit the logistic model to your data and used the MLEs to estimate $\tilde{p}$ for those animals that weren't telemetered. So, it still estimates the same number of parameters, it just doesn't use those parameters to estimate $\tilde{p}$ for some individuals. Thus, it doesn't make sense to compare the 2 approaches using AIC_c to see which is a more appropriate. It's completely up to the user and you should make a decision as to which way you'd like to go before you start running models, then run each model the same way.

Model	AICc	Delta AICc	AICc Weight	Model Likelihood	No. Par.	Deviance	-2Log(L)
{p(DTE)p~(DTE)}	351.0997	0.0000	0.50000	1.0000	4	342.8416	342.8416
{p(DTE)p~(DTE) observed}	351.0997	0.0000	0.50000	1.0000	4	342.8416	342.8416
{p(.)p~(site)}	401.1442	50.0445	0.00000	0.0000	3	394.9904	394.9904
{p(.)p~(.)}	405.2051	54.1054	0.00000	0.0000	2	401.1286	401.1286
{p(site)p~(.)}	405.5486	54.4489	0.00000	0.0000	3	399.3948	399.3948

You might argue that since we're making inference using a model-based approach, we should be consistent and apply the model to all individuals. Or, you might argue that there is no sense in estimating $\tilde{p}$ from a model when you can estimate it directly via sampling. Either approach is defensible. In practice, this decision often has little consequence (especially if nearly all animals are telemetered) and estimates are similar regardless of which flavor of model you prefer. For example, if we compare the derived density estimates from the model we just ran to `p(DTE)p~(DTE)` we see that the estimates change slightly but we would probably make a similar inference regardless of the approach.

Estimates of Derived Parameters
Density Estimates of {p(DTE)p~(DTE) observed p~}

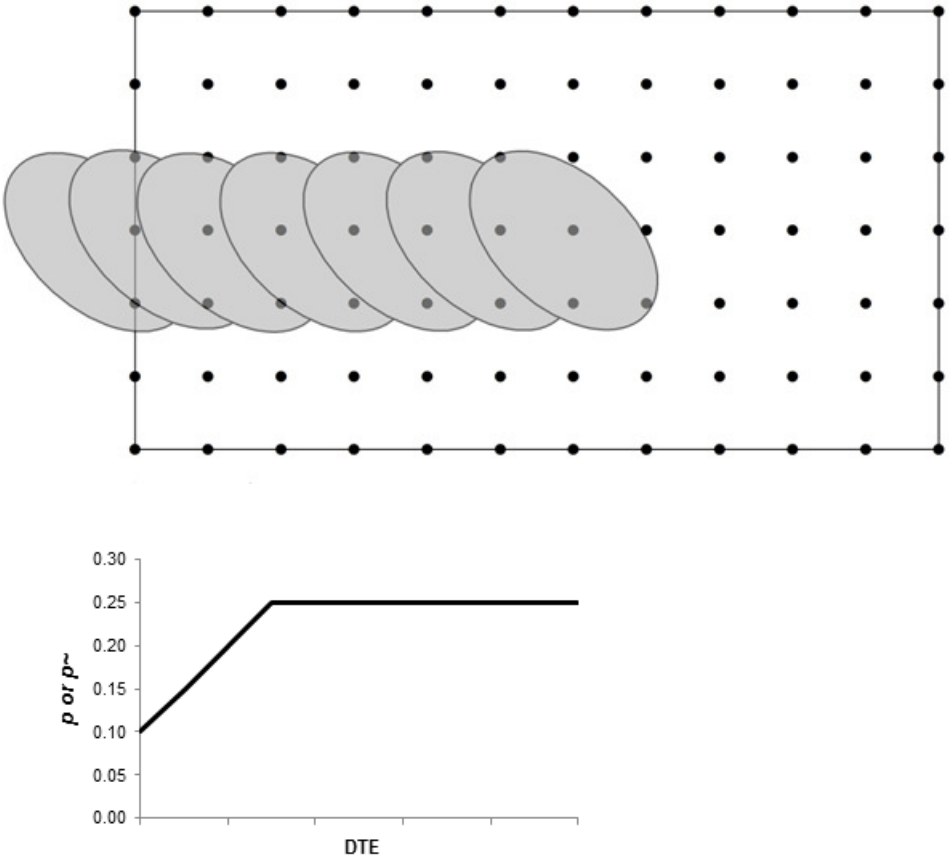
Group	D-hat	Standard Error	95% Confidence Interval Lower	Upper
1	27.213095	9.3592483	13.868310	53.398903
2	14.686639	5.9329326	6.6536185	32.418055

Estimates of Derived Parameters
Density Estimates of {p(DTE)p~(DTE)}

Group	D-hat	Standard Error	95% Confidence Interval Lower	Upper
1	25.280231	9.0758852	12.507905	51.094896
2	16.683452	6.1409217	8.1089753	34.324629

20.2.2. Threshhold Model

Depending on the relationship between the size of your study site and the home range size of your target species, you may find that you only want the DTE covariate to operate up to a certain threshold. Consider the following example in which home ranges are now smaller than they were in our initial figure above.



As we move through the home ranges from left to right, once we encounter the 3rd or 4th home range, those animals are fully on the site and exposed to the same number of traps as those farther to the right (more toward the center of the site). In this situation, p and/or $\tilde{p}$ may increase with DTE for a given distance, but after that, DTE is no longer a very good predictor of either parameter. Thus, it may be useful to consider a "threshold model" for the estimation of $\tilde{p}$ and/or p such that the DTE covariate is only meaningful to a point. After that p and/or $\tilde{p}$ are constant.

Mathematically, such a model can be represented as

$$\text{logit}(\hat{\tilde{p}}_i) = \hat{\beta}_1 + \hat{\beta}_2\left(\min(\hat{\beta}_3, \text{DTE})\right)$$

where $\hat{\beta}_1$ and $\hat{\beta}_2$ are the usual intercept and slope estimates from a logistic model using distance to edge as a covariate, and $\hat{\beta}_3$ is the threshold parameter (i.e., the point at which the relationship between p and/or $\tilde{p}$ asymptotes). Note however, that you cannot build a model exactly like this in **MARK**. **MARK** is unable to estimate a parameter that is embedded within the '`min`' function. Instead, if you decide that a threshold model may be appropriate for your situation, build several models in which you give **MARK** an actual threshold value for DTE, then use AIC_c to help you figure out what the threshold should be.

In the example data set we've been analyzing, the trapping grid was (10 × 10) with 10m spacing. Thus, the center of the grid is 45m from any edge when the edge of the grid is considered to be exactly at the trap positions at the outer edge of the grid. Note that distance to the traps at the corners is in fact $(\sqrt{2} \times 45)$ from the center of the grid. Given that the edge of the grid is taken as the trap position on the outer edge, the threshold values we consider should be less than 45 (hopefully you can see that if we specify a model with a threshold of 45 in this specific example, we haven't specified a threshold at all – p or $\tilde{p}$ increase with DTE all the way to the center of the grid). Likewise, at the other end of the spectrum, we expect the threshold to be greater than at least the 1st trap width (remembering there should be 4 traps per home range). Thus, let's provide **MARK** with possible threshold values of 15, 25, and 35m for the p parameter. To do this, simply translate the equation above into a design matrix, making use of the design matrix function '**`min`**' (see Chapter 11). Thus, the design matrix for a threshold of 15m looks like:

B1: p int	B2: p DTE 15m	Parm	B3: p~ int	B4: p~ DTE
1	min(15,DTE)	1:p	0	0
1	min(15,DTE)	2:p	0	0
1	min(15,DTE)	3:p	0	0
1	min(15,DTE)	4:p	0	0
1	min(15,DTE)	5:p	0	0
1	min(15,DTE)	6:p	0	0
1	min(15,DTE)	7:p	0	0
1	min(15,DTE)	8:p	0	0
1	min(15,DTE)	9:p	0	0
1	min(15,DTE)	10:p	0	0
1	min(15,DTE)	11:c	0	0
1	min(15,DTE)	12:c	0	0
1	min(15,DTE)	13:c	0	0
1	min(15,DTE)	14:c	0	0
1	min(15,DTE)	15:c	0	0
1	min(15,DTE)	16:c	0	0
1	min(15,DTE)	17:c	0	0
1	min(15,DTE)	18:c	0	0
0	0	19:ptilde	1	DTE
0	0	20:ptilde	1	DTE

For other thresholds, just replace the '15' in the design matrix expression with a different number (25 or 35). If we examine the results browser (top of the next page) we see that the 15m threshold is the best fitting model of the set we've constructed thus far.

Model	AICc	Delta AICc	AICc Weight	Model Likelihood	No. Par.	Deviance	-2Log(L)
{p(DTE 15m)p~(DTE)}	348.1651	0.0000	0.50762	1.0000	4	339.9071	339.9071
{p(DTE 25m)p~(DTE)}	350.7230	2.5579	0.14129	0.2783	4	342.4649	342.4649
{p(DTE)p~(DTE)}	351.0997	2.9346	0.11703	0.2305	4	342.8416	342.8416
{p(DTE)p~(DTE) observed p~}	351.0997	2.9346	0.11703	0.2305	4	342.8416	342.8416
{p(DTE 35m)p~(DTE)}	351.0997	2.9346	0.11703	0.2305	4	342.8416	342.8416
{p(.)p~(site)}	401.1442	52.9791	0.00000	0.0000	3	394.9904	394.9904
{p(.)p~(.)}	405.2051	57.0400	0.00000	0.0000	2	401.1286	401.1286
{p(site)p~(.)}	405.5486	57.3835	0.00000	0.0000	3	399.3948	399.3948

20.3. Confidence Intervals

Recall from Chapter 14 that **MARK** computes log-based confidence intervals for abundance, partly due to lack of asymptotic normality in point estimates and partly due to the idea that we want the lower bound to be no less than M_{t+1} (makes no sense to have a lower confidence limit that is smaller than the number of animals you know are out there!). Superficially, we'd like to invoke a similar approach with density, and for the same reasons. Notice in section 14.1.10 that M_{t+1} (the number of animals capture or the minimum bound on the abundance estimate) appears in the calculations as does f_0, which is simply $\hat{N} - M_{t+1}$. You may be tempted to conclude that the analog to M_{t+1} for density estimation would the sum of all the $\tilde{p}_i$ divided by the area of the study site. Then the analog to f_0 would be $\hat{D} - (\sum \tilde{p}_i)/A$ and all of the same computations would follow. However, $\sum \tilde{p}_i$ is not a known quantity as M_{t+1} was in the abundance world. It's just an estimate (i.e., it's really $\sum \hat{\tilde{p}}_i$). Thus, instead of following the closed capture CI computation exactly, **MARK** uses log-based confidence intervals that ensure the CIs are > 0, which is the minimum condition that has to be true. As such the 95% CIs are computed as:

$$\pm \exp\left[\log(\hat{D}) \pm 1.96\left(\frac{SE(\hat{D})}{\hat{D}}\right)\right]$$

20.4. Assumptions

Recall from Chapter 14 (and earlier in this chapter) that closed capture models require 2 assumptions: (1) the population is closed geographically (animals do not move on and off of the study site) and (2) demographically (there are no births, deaths, immigration or emigration). Through the use of auxiliary telemetry data we are able to relax the first of these assumptions when estimating density, but we still need the population to be closed demographically. Fortunately, demographic closure is usually much easier to attain and simply requires sampling for a relatively short duration at a time when you don't expect births or dispersal events. However, the density estimation model described here requires 3 additional assumptions beyond the usual closed capture ones:

1. The animals sampled with telemetry are representative of the population of animals that use the study site.
2. Telemetry devices do not affect movements and there are no effects of mark-recapture sampling on animal movements.
3. Error in telemetry locations is small relative to the size of the study site and assignment (on/off) of locations near the edge of the site is unbiased.

The crux of assumption 1) is that you want to avoid oversampling "interior" animals while under-sampling "edge" animals or vice-versa. Hopefully if you think back the form of the estimator, you can see why either of these cases would be problematic. If your severely under-sample edge animals

(which should have relatively small values for $\tilde{p}_i$), the summation of $\tilde{p}_i$ in the numerator of our density expression will be too large (all we sampled were animals with $\tilde{p}_i$ near 1.0), and you will end up with an estimate of density that is biased high to some degree. The reverse could also be true, and would result in negatively biased results, although it is probably more likely that you would capture too many interior animals compared to too many edge animals. Note, however, we've also previously determined that DTE is often a good predictor of capture probability. That is, interior animals may well have higher capture probabilities (p) than edge animals because they tend to have more traps or detectors in their home range. If this relationship holds in your study then it may be true that you tend to sample interior animals too often, which inflates the numerator of our density expression, but we're probably inflating the denominator at the same time, thereby canceling at least some of the potential bias. Nevertheless, you want to provide yourself with as much protection from bias as possible, and there are a couple of design features of mark-recapture sampling that may help.*

For assumption 2, the main concern is if baits or lures were used to detect animals during the mark-recaptrue portion of your study, which is commonplace. Such attractants are designed to alter the normal movements of animals (hopefully they entice animals to your detectors and increase your capture probability!), but when we're sampling animal movements using telemetry we'd like animal movements to be "natural". We have several recommendations for telemetry sampling when baits or lures were used during mark-recapture.

First, we recommend discarding from the analysis any telemetry locations obtained during the mark-recapture session as it is very probable that baited detectors influenced animal movement. Second, it is imperative that you the investigator ensure that every last morsel of bait has been removed from the site at the end of the mark-recapture session so there is no unnatural attractant left to influence movements after the session. Third, we suggest that it may be appropriate to wait 1–2 days post mark-recapture before collecting location data to allow animals to revert to their normal activity patterns. Telemetry sampling should, however, be completed within a reasonable time to avoid biasing estimates of $\tilde{p}_i$ due to seasonal movements, migration, or dispersal. In summary, our experience suggests that bait and lures can have quite an impact on the movements of animals during a mark-recapture study, even enticing animals to make extensive movements to become part of the mark-recapture data set when their usual home range does not include the study site at all (e.g., animal 4 in the original example). Following the guidelines outlined here, telemetry information should result in $\tilde{p}_i = 0$ for these individuals, and they will not contribute to the density estimate, which is appropriate.

20.5. Summary

The summary for Chapter 14 states that "Despite a seemingly simple goal, estimating abundance can be quite difficult...". The same can be said for density estimation due to all of the numerous, subtle complications embodied in the closed capture models along with some added complications necessary to convert abundance to density. Despite these difficulties, the "**`Density with Telemetry`**" data type in **MARK** provides a way forward and does so using auxiliary information that actually measures animal

* For traditional live-trapping studies in which telemetry devices are deployed on a subset of animals during the mark-recapture sampling, we recommend checking traps in the same order on each occasion, but selecting a different random starting point each time. This strategy should help equalize the probability of capturing and telemetering edge vs. interior animals. In addition, we suggest retaining some telemetry devices for deployment during the latter portion of a sampling session to facilitate the inclusion of trap-shy individuals in the radio-tagged sample in addition to trap-happy individuals that are captured early and often.

All of this hand waving is also a justification for using the logistic regression model for $\tilde{p}$ rather than the observed values, because the logistic regression model would correct for the bias of a non-random sample of animals that were radioed, assuming that at least a few edge animals (if not a random sample) made it into the telemetry sample.

movement on and off of the study site, which is the source of the density estimation problem. There are other contemporary means of computing density as well, and we encourage readers to explore Ivan *et al*. 2013 for a discussion of advantages and disadvantages of this approach compared to others.

20.6. References

Anderson, D. R., K. P. Burnham, G. C. White, and D. L. Otis. 1983. Density estimaton of small mammal populations using a trapping web design and distance sampling methods. *Ecology*, **64**, 674-680.

Borchers, D. L., and M. G. Efford. 2008. Spatially explicit maximum likelihood methods for capture-recapture studies. *Biometrics*, **64**, 377-385.

Converse, S. J., G. C. White, and W. M. Block. 2006. Small mammal responses to thinning and wildfire in ponderosa pine-dominated forests of the southwestern United States. *Journal of Wildlife Management*, **70**, 1711-1722.

Dice, L. R. 1938. Some Census Methods for Mammals. *Journal of Wildlife Management*, **2**, 119-130.

Efford, M. 2004. Density estimation in live-trapping studies. *Oikos*, **106**, 598-610.

Efford, M. G., D. L. Borchers, and A. E. Byrom. 2009. Density estimation by spatially explicit capture-recapture: likelihood based methods. Pages 255-269 in G. P. Patil, D. L. Thomson, E. G. Cooch, and M. J. Conroy, editors. *Modeling Demographic Processes in Marked Populations*. Springer Science, Boston, Massachusetts, USA.

Huggins, R. M. 1989. On the statistical analysis of capture-recapture experiments. *Biometrika*, **76**, 133-140.

Huggins, R. M. 1991. Some practical aspects of a conditional likelihood approach to capture experiments. *Biometrics*, **47**, 725-732.

Ivan, J. S., G. C. White, and T. M. Shenk. 2013. Using simulation to compare methods for estimating density from capture-recapture data. *Ecology*, **94**, 817-826.

Kendall, W. L., J. D. Nichols, and J. E. Hines. 1997. Estimating temporary emigration using capture-recapture data with Pollock's robust design. *Ecology*, **78**, 563-578.

Link, W. A., and R. J. Barker. 1994. Density estimation using the trapping web design: a geometric analysis. *Biometrics*, **50**, 733-745.

Otis, D. L., K. P. Burnham, G. C. White, and D. R. Anderson. 1978. Statistical inference from capture data on closed animal populations. *Wildlife Monographs*, 7, 135.

Parmenter, R. R., T. L. Yates, D. R. Anderson, K. P. Burnham, J. L. Dunnum, A B. Franklin, M. T. Friggens, B. C. Lubow, M. Miller, G. S. Olson, C A. Parmenter, J. Pollard, E. Rexstad, T. Shenk, T. R. Stanley, and G. C. White. 2003. Small-mammal density estimation: a field comparison of grid-based vs. web-based density estimators. *Ecological Monographs*, **73**, 1-26.

Royle, J. A., and R. M. Dorazio. 2008. Conceptual and philosophical considerations in ecology and statistics. Pages 444 in J. A. Royle, and R. M. Dorazio, editors. *Hierarchical modeling and inference in ecology*. Academic Press, Boston, Massachusetts, USA.

Royle, J A., J. D. Nichols, K. U. Karanth, and A. M. Gopalaswamy. 2009. A hierarchical model for estimating density in camera-trap studies. *Journal of Applied Ecology*, **46**, 118-127.

Schwarz, C. J., and A. N. Arnason. 1996. A general methodology for the analysis of capture-recapture experiments in open populations. *Biometrics*, **52**, 860-873.

White, G. C., and T. M. Shenk. 2001. Population estimation with radio-marked animals. Pages 329-350 in J. J. Millspaugh, and J. M. Marzluff, editors. *Radio tracking and animal populations.* Academic Press, San Diego, California, USA.

Williams, B. K., J. D. Nichols, and M. J. Conroy. 2002. *Analysis and Management of Animal Populations*. Academic Press, New York, New York.

Wilson, K. R., and D. R. Anderson. 1985. Evaluation of two density estimators of small mammal population size. *Journal of Mammalogy*, **66**, 13-21.

Zahratka, J. L., and T. M. Shenk. 2008. Population estimates of snowshoe hares in the southern Rocky Mountains. *Journal of Wildlife Management*, **72**, 906-912.

APPENDIX A

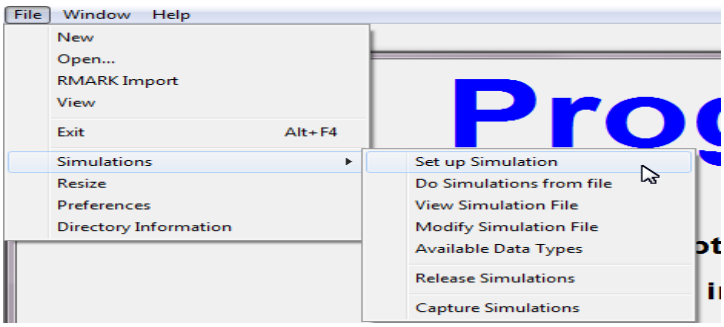
Simulations in MARK . . .

The ability to simulate data and fit models for various data types (e.g., Cormack-Jolly-Seber, multi-state, and so on) is very useful, for a variety of purposes. While it is possible to write your own simulation code (which has the advantage of making you acutely aware of the underlying structure of the model), **MARK** has a convenient and very powerful built-in simulation capability.

At present, **MARK** can simulate data under three primary systems: program **RELEASE** simulations, program **CAPTURE** simulations, and simulation of models developed in program **MARK**. In this appendix, we'll focus on the simulation of data for two common data types – a simple CJS design, using both **MARK** and **RELEASE** simulations.

A.1. Simulating CJS data

To simulate CJS data in **MARK**, start **MARK**, and select '**`File | Simulations`**'. At this point, you'll see that you are given several options of how you want to simulate the data:

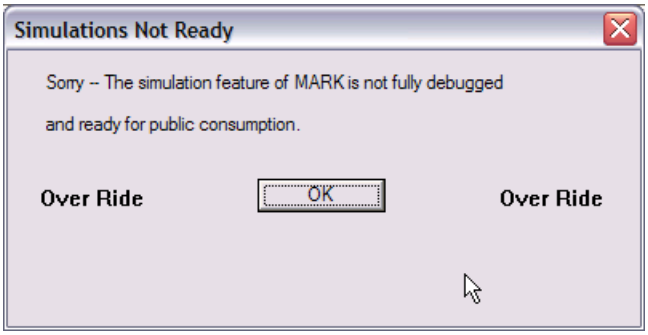


Notice that one of the options presented in the drop-down menu is for '**`Release Simulations`**'. You might recall that program **RELEASE** is appropriate for CJS data, which is what we're going to try to simulate here. So, in fact, for this example, we could select the '**`Release Simulations`**' options from the menu. However, here we introduce the more general approach to simulating data in **MARK**, which can be used for data types other than CJS (we will revisit **RELEASE** simulations in a moment).

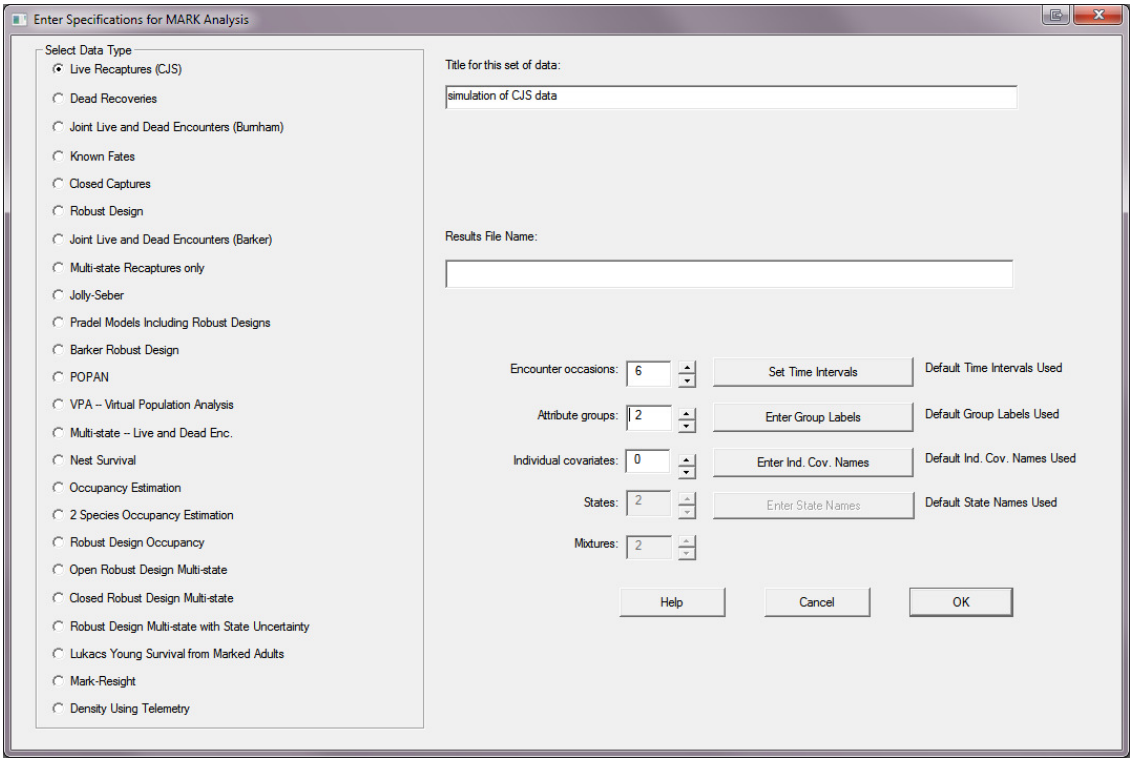
In addition, one of the selections in the '**`Simulation`**' menu will generate a list of the '**`data types`**' in **MARK** under which you can simulate data. Most (but not all) data types can be simulated.

A.1.1. Simulating CJS data - MARK simulations

To simulate CJS data in **MARK**, we need to select '**`Set up Simulation`**' from the drop-down menu. Selecting this option in **MARK** will generate a puzzling popup window:



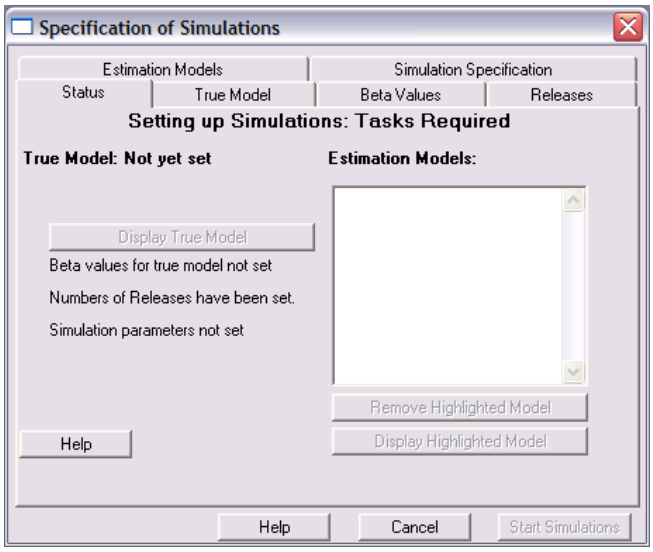
This window is trying to warn you that not all elements of the simulation feature in **MARK** are fully debugged. If you try to click the '**`OK`**' button, a little '**`OverRide`**' message will keep popping up, and jumping from side to side of the window as you attempt (in vain) to click on it. However, if you double-click anywhere in the grey area of the window (not the '**`OK`**' button), you are in fact able to over-ride the warning window, and will (finally) be presented with essentially the **MARK** specification window, which you're probably quite familiar with by now.



It's at this point we specify the data type we want to simulate - in this case, CJS '**`recaptures only`**'

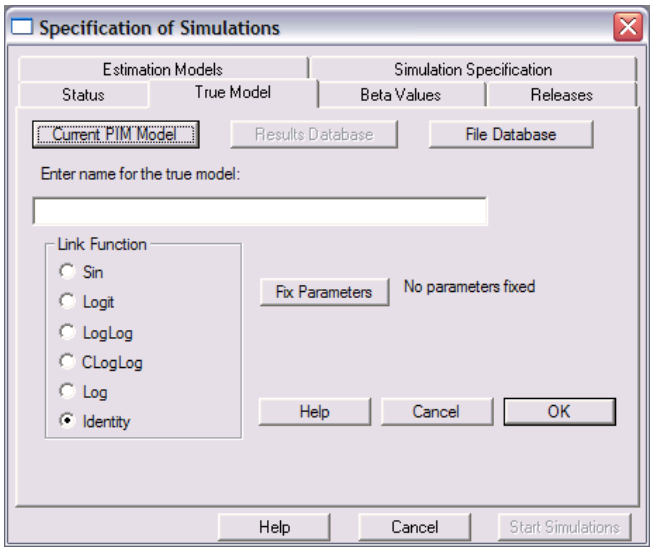
data. Let's simulate 2 groups, 6 occasions. Note that we've entered a title for the simulation, but have left the box for '**Results File Name**' blank - since neither are needed.

Once you click the '**OK**' button, you'll be presented with the main simulation setup window.



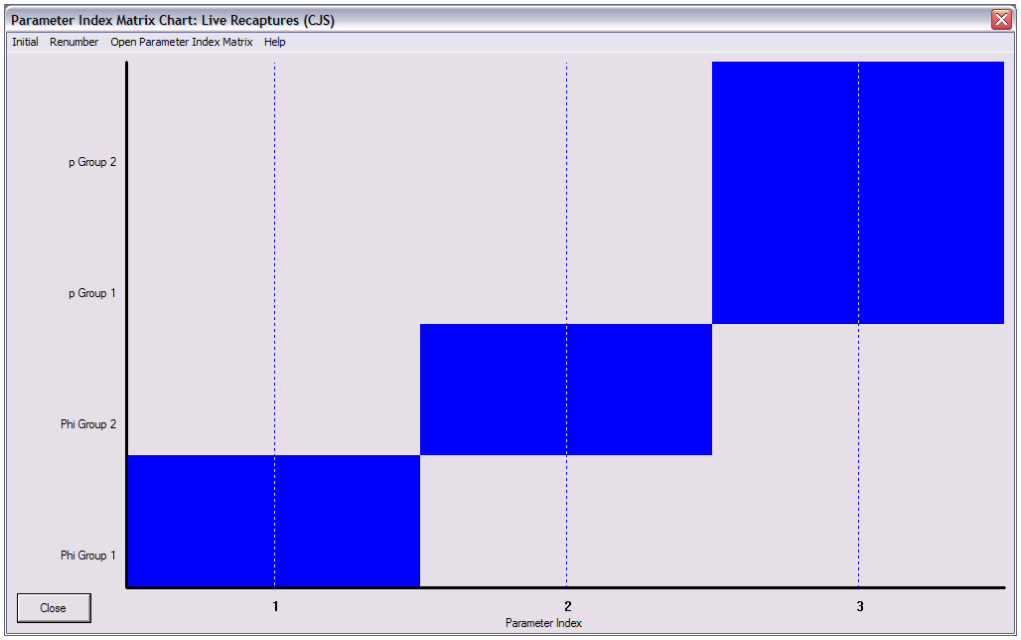
The window consists of a series of tabs, each of which correspond to a particular aspect of the simulation that needs to specified before you can run the simulation. It's easiest to go through the various steps sequentially, corresponding to each of the tabs in the simulation setup window.

The first step is to specify the '**True Model**', by selecting that tab.



Now, we need to specify the parameter structure of the 'true model' we want to simulate. For purposes of demonstration, we'll use model - $\{\varphi_g p_.\}$ - differences between the two groups for φ, but no time variation, and no group or time differences for p. So, the first thing we need to do is set up the

appropriate parameter structure - this is most easily accomplished by manipulating the PIM chart. By now this should be pretty familiar territory.



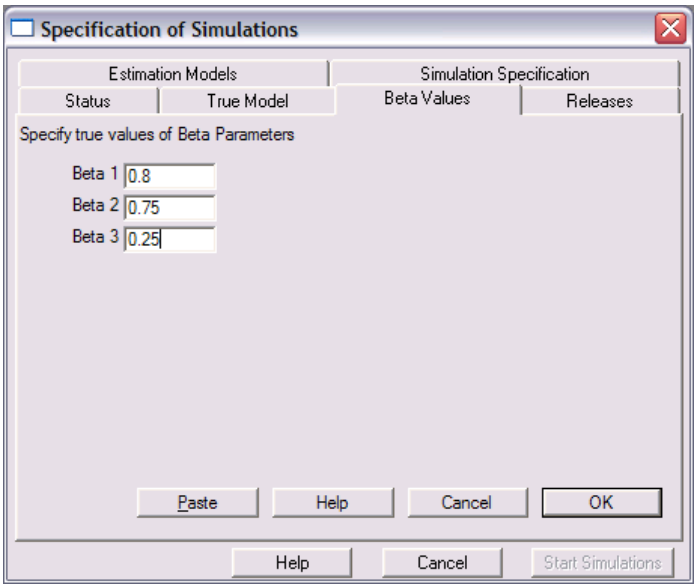
Several important things to notice here before we proceed. First, as shown on the preceding page, we've specified the identity link (rather than the sin or logit link) - this is important since we want to enter the parameter values for our model on the real scale (this is simply for convenience - we could of course calculate the sin or logit transformed values for each parameter, if we so chose). Next, notice that there are 3 radio buttons just above the title box: one (left-most) is for '**`Current PIM Model`**'. The next (greyed-out) is for the '**`Results Database`**'. Finally, right-most, is the '**`File Database`**'. For the moment, the one we're interested in is the the first one - we click '**`Current PIM Model`**', so that **MARK** will know to use the parameter structure we just created (corresponding to $\{\varphi_g p_.\}$), for our simulation. When you click this button, the title will be replaced momentarily with '**`???`**'. No worries - simply enter the title for your true model (you might use '`true model`', or something more explicit). Then, click the '**`OK`**' button. Notice that now the button '**`Display True Model`**' is now active (it was previously greyed-out). This button allows you to check the true model, if you want.

Next, we select the '**`Beta values`**' tab - here is where we put in the scale-appropriate β values for the linear model corresponding to our model. Since we specified the identity link, all we need to do is specify the parameter values on the real scale. For this example, we'll use the following parameter values: $\varphi_{grp\,1} = 0.80$, $\varphi_{grp\,2} = 0.75$, $p = 0.25$. So, a difference of 0.05 in apparent survival between the two groups, and a (relatively) low detection probability. So, we might be interested in how large a sample size of newly marked individuals we need to release on each occasion in order to allow us to detect a difference in survival of this magnitude.

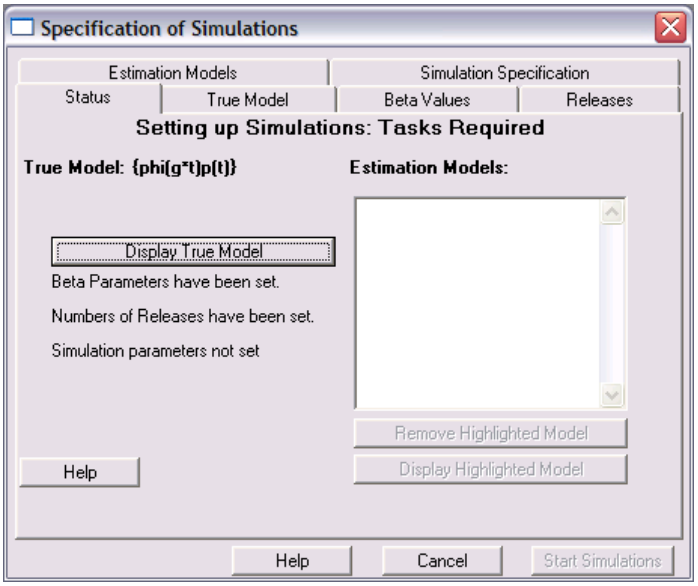
So, all that we need to do is enter these β values into **MARK**. These values are used with the the design matrix and the link function to produce the value of the real parameters. A good choice for the link function to be used with an identity design matrix is the *identity link function*, because then the values for the β parameters are the same as for the real parameters. For other link functions, the β parameters must be set to values that will produce the correct value of the real parameter via the link function. There is, however, an exception when the β values entered define the exact parameter being

estimated. This issue occurs with simulation of the robust design models (see section A.4).

For the moment, though, we can proceed using the identity link function. Simply click the '**Beta Values**' tab, and enter the parameter values.



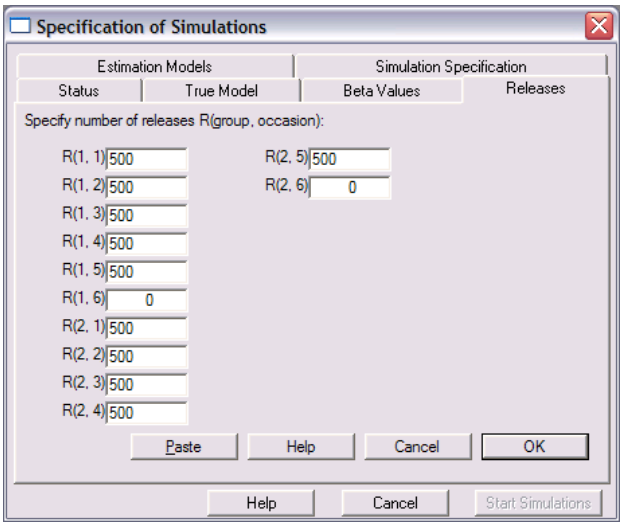
Once you've entered the appropriate values, and clicked the '**OK**' button, you'll be popped back to the main part of the setup window.



Notice that **MARK** conveniently provides you with visual indications of which steps in setting up the simulations you've completed (never let it be said that **MARK** isn't user-friendly!).

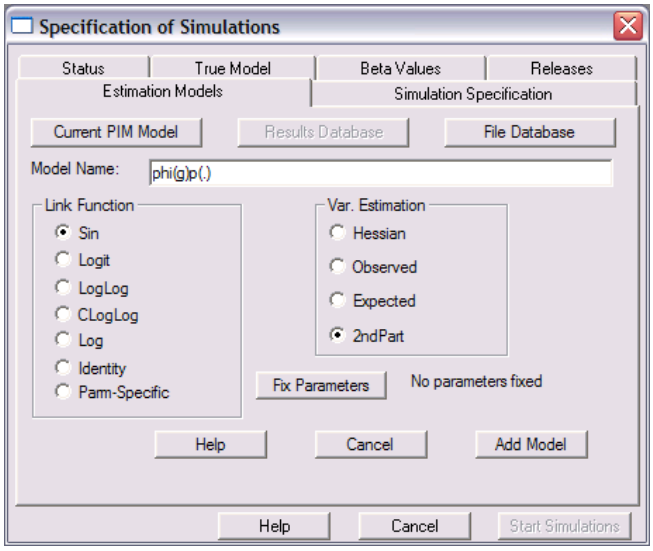
Next, we want to specify the number of releases of newly marked individuals at each occasion. So,

we click the '**`Releases`**' tab. For this simulation, we'll try 4 different 'release experiments' - to explore the influence of sample size on the ability to detect true differences in survival between the two groups. So, we'll try releasing 500, 250, 100, and 50 newly marked individuals at each occasion, respectively. We'll need to do this one 'experiment' at a time - here is what we would enter for the '500' sample size simulation:



Notice we don't release new individuals on the last occasion, since these individuals will provide no information (given that the study terminates on the last occasion).

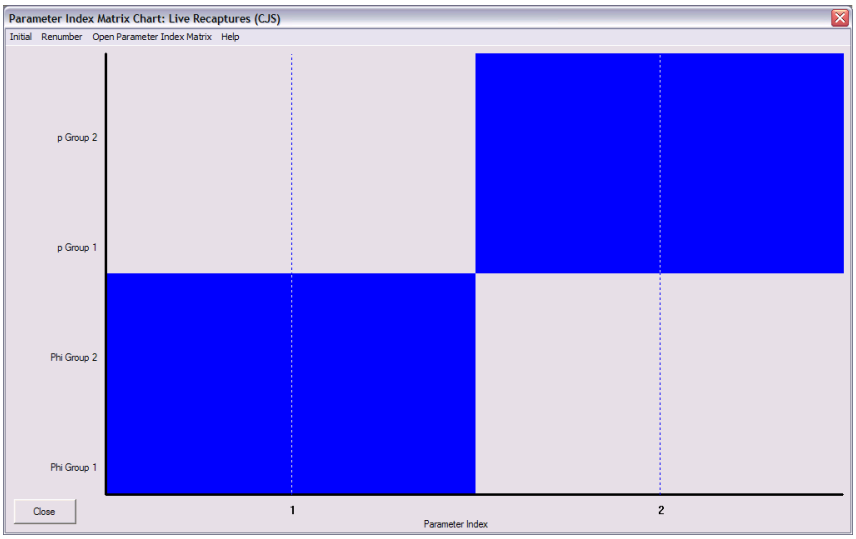
Next, we want to specify the models we want to fit to the simulated data. To do this, click the '**`Estimation Models`**' tab.



Here, we could simply simulate the current (true) model, by clicking the '**`Current PIM Model`**' button, or we could specify another model. If we choose another model, we need to specify the PIM structure for

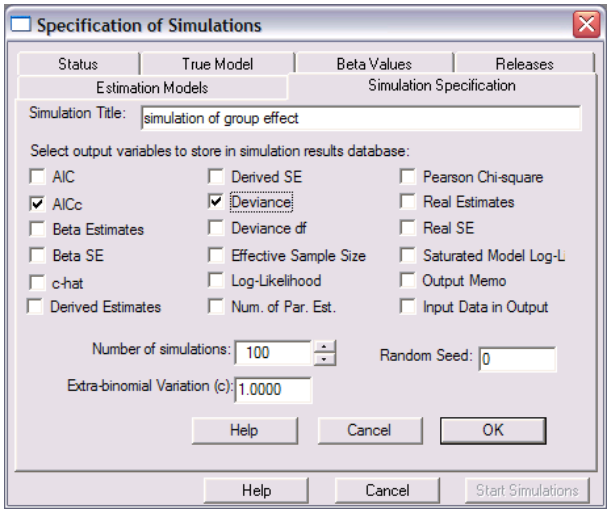
the model, in the usual way. For our experiment, we want to compare fitting the true model $\{\varphi_g p_.\}$ with a reduced parameter model with no group effect on survival $\{\varphi_. p_.\}$. Since the true model is reflected by the current PIM structure, simply click the '**`Current PIM Model`**' button. Enter '`phi(g)p(.)`' as the model name. Then, specify the link function. What you're doing here is telling **MARK** which link function you want to use during the numerical estimation for a given simulated data set. Usually, we would use either the sin or logit link.

Once finished, click the '**`Add Model`**' button. Now, we want to add the reduced parameter model $\{\varphi_. p_.\}$. To do this, we simply need to select the '**`Estimation Models`**' tab again, open up the PIM chart, and change the parameter structure to reflect this model.



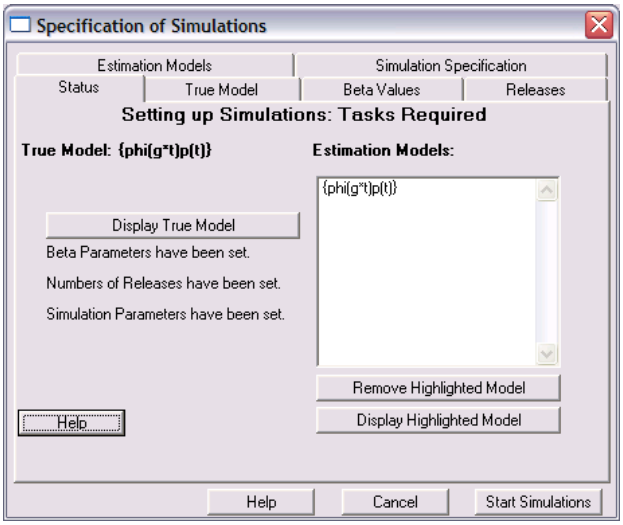
Once you've modified the PIM chart, simply click the '**`Current PIM model`**' button, enter '`phi(.)p(.)`' as the model title, and select the appropriate link function. Then, click the '**`Add model`**' button.

Finally, we're ready to specify the simulation - click the '**`Simulation Specification`**' tab.



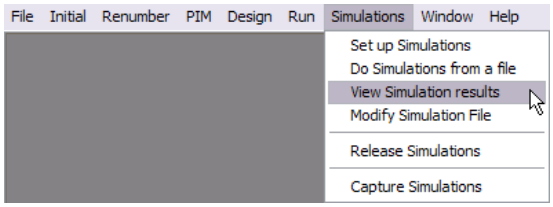
Here, you can specify the various statistics you want to output, how many simulations you want to run, and whether or not you want to add extra-binomial noise to your data (i.e., specify a specific, true value for c). For now, we'll run 100 simulations, and output both the AIC_c and the deviance for each model for each of the simulated data sets.

Once you've completed specifying the simulations, click the '**OK**' button. This will bring you back to the main simulation window.

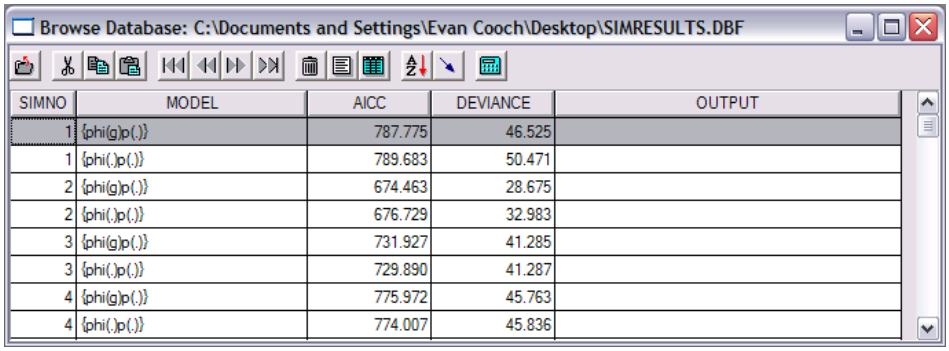


At this point, all that is left is to run the simulations. Simply click the '**Start Simulations**' button - **MARK** will then ask you where you want to store the simulation output. Once you've done that, **MARK** will starting grinding through however many simulations you've specified – clearly, the length of time this takes is dependent upon the model(s) you are simulating, the number of releases at each occasion, and how 'hot-rod' your computer is. **MARK** will present a progress bar as it works through each simulation.

Once completed, you need to 'do something' with the simulation results. While for our experiment we will ultimately want to extract summary data from the output, for the moment, we'll explore a couple of different ways to view the simulation results. To do so, select the '**View Simulation results**' option from within **MARK**.

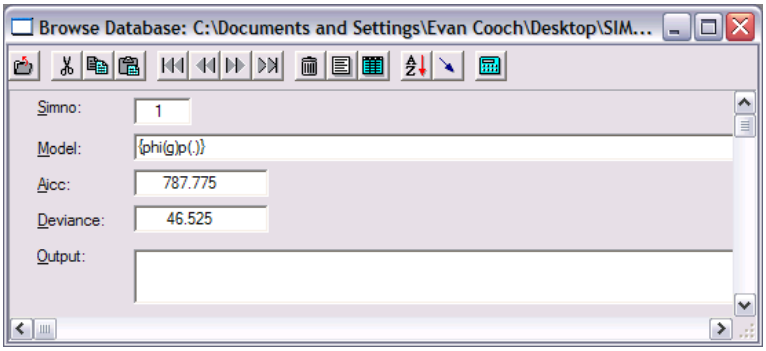


MARK will present you with a database tabulation of the simulation results - model name(s), and any of the statistics you requested (in this case '**AICC**' (i.e., AIC_c), and '**DEVIANCE**'). See the figure at the top of the next page.



SIMNO	MODEL	AICC	DEVIANCE	OUTPUT
1	{phi(g)p(.)}	787.775	46.525	
1	{phi(.)p(.)}	789.683	50.471	
2	{phi(g)p(.)}	674.463	28.675	
2	{phi(.)p(.)}	676.729	32.983	
3	{phi(g)p(.)}	731.927	41.285	
3	{phi(.)p(.)}	729.890	41.287	
4	{phi(g)p(.)}	775.972	45.763	
4	{phi(.)p(.)}	774.007	45.836	

The 'view' we're looking at is known as the '*browse view*'. There is another view known as '*form view*', which has some advantages we'll explore later on. To access 'form view', simply select the '**`Form View`**' icon on the toolbar (or by puling down the '**`View`**' menu and selecting the '**`Form View`**' option). Again, we'll discuss further uses of this view later, but for now, here's what it looks like at least – simply select a particular record from the browser, and then select '**`Form View`**':



For our experiment, we're interested in how much sample size affects our ability to detect a difference in survival between the two groups. Recall that in our true model, the difference was 0.05 (0.80 vs 0.75). There are a number of ways we can approach this question, using the results of our analysis of the simulated data, but for simplicity, we'll invoke the classical likelihood ratio test (LRT) approach (if you forget the basics of the LRT, see chapter 4). For each of the 100 simulations, we'll calculate the LRT test statistic, and assess whether or not the difference in model deviances (between the true and reduced parameter models) is significant at the $\alpha = 0.05$ level, for each of the 4 simulated sample sizes. For purposes of comparison, we'll also present mean ΔAIC_c and evidence ratios and model likelihoods for each sample size. Recall that the evidence ratio of a given model relative to the best model is given as

$$\frac{w_1}{w_j} \equiv \frac{1}{e^{-1/2\Delta_j}} \equiv e^{1/2\Delta_j}$$

and the model likelihood (i.e., the probability that model $\{\varphi\, p_{\cdot}\}$ is the K-L best model) is simply the inverse of the evidence ratio (i.e., w_j/w_1). Note that you can't do these calculations directly with **MARK** – the simulation results DBF file must be 'imported' into some external spreadsheet or statistics program to allow you to generate summary statistics; e.g., calculate and tabulate LRT statistics.

sample size	*% significant*	*mean ΔAIC*	*evidence ratio*	*likelihood*
50	16	1.78	2.44	0.410
100	37	2.76	3.97	0.252
250	71	5.10	12.81	0.078
500	94	11.53	318.94	0.003

What is pretty clear from this table is that for a release (sample) size <250, there is a low probability of detecting a real survival difference of 0.05, given an encounter probability of $p = 0.25$. Only when >250 individuals are newly marked at each occasion is the probability of detecting a true difference of 0.05 in survival ~ 95%. These conclusions are supported by the evidence ratios and model likelihoods. Note that for sample sizes < 500, the likelihood for model $\{\varphi\, p_{\cdot}\}$ is > 0.05, meaning that based on a nominal error rate of $\alpha = 0.05$, you would have no basis for 'rejecting' model $\{\varphi\, p_{\cdot}\}$, even though it is false. Only when sample size is > 250 is the likelihood of model $\{\varphi\, p_{\cdot}\}$ being K-L best < 0.05.

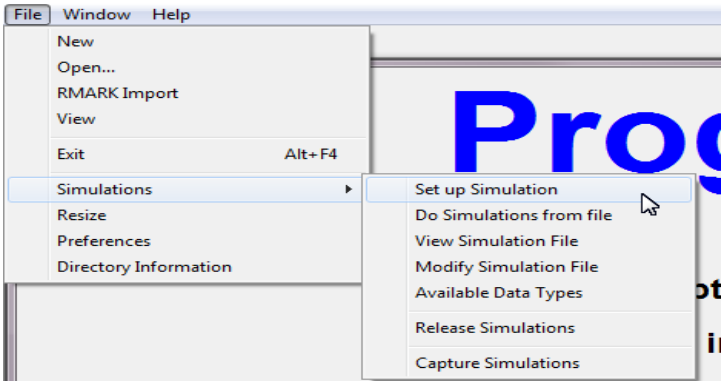
Of course, you may be familiar with many projects that would be hard-pressed to release even 100 newly marked individuals at each occasion. So, in such cases, your only option is to increase the encounter probability p (this is generally known as 'the big law' - in general, you should do everything possible to increase encounter rate). Failing that, you'll have to accept that there will be real limits to the power of the inference you can make from your data.

The preceding is an example of using the simulation capabilities in **MARK** to perform what is in effect a 'power analysis' - this could (and should) be done prior to a study to help evaluate the amount of 'effort' your study will require in order to achieve an inference of a given precision. Of course, in this example we've assumed a particular 'true model' - the results of your simulations will clearly be influenced by the choice of true models.

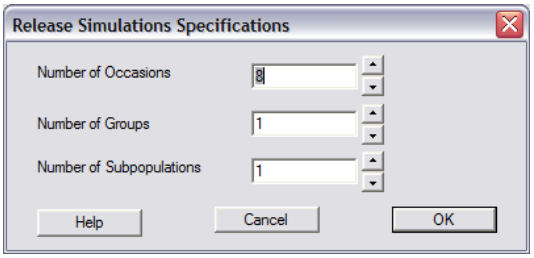
A.1.2. Simulating CJS data - RELEASE simulations

For this example, we'll look at simulating data under program **RELEASE** - to demonstrate that $\hat{c}$ is in fact an estimate of some underlying parameter, c (and as such, is estimated with uncertainty). This is important when using estimates of $\hat{c}$ for quasi-likelihood adjustments of AIC_c (see chapter 5).

To simulate CJS data under program **RELEASE**, simply start up **MARK**, and select '**`File | Simulations`**'. At this point, you'll see that you are given several options of how you want to simulate the data:



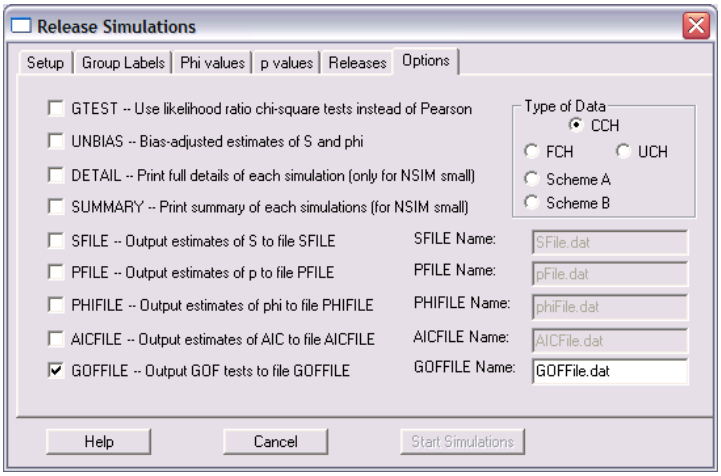
Notice that one of the options presented in the drop-down menu is for '**Release Simulations**'. Select this option. This will bring up a smaller window where you will be asked to specify the number of release occasions, the number of groups, and number of subgroups. For this example, we want 8 occasions, and 1 group (and 1 subgroup):



Once you click the '**OK**' button, you'll be presented with yet another tabbed window, where each tab corresponds to a specific parameter or option. The default is for 1000 simulations, with extra binomial variation of 1 (corresponding to $\hat{c}$=1), and a random number seed of zero (meaning some random function of the computer clock will be used to seed the simulations). This is shown at the top of the next page. All you need to do at this stage is answer/complete each of the 'tabbed windows'.

Start by setting a group label, then setting value for φ and p respectively, the number of releases on each occasion, and some options. For our simulations, we want some time variation in both φ and p, with 250 releases on each occasion. For example, we'll use some random values between 0.5 and 0.75 for both parameters.

For the '**options**' tab, we want to output the GOF test statistics for each simulated data set (all 1000 of them, in this case), so we check the **GOFFILE** box. You may recall from some long-ago statistics class that you can use either a Pearson χ^2 approach to contingency analysis, or a likelihood-based G-test approach. The simulation option windows lets you choose either. For the moment, we'll use the Pearson χ^2 approach, so leave the **GTEST** box unchecked:

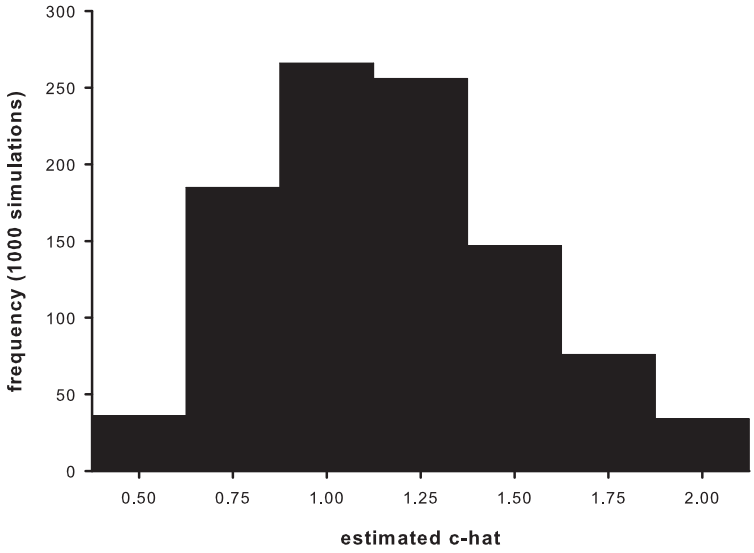


In the upper right corner of the options window are several radio-buttons corresponding to different data types. CCH refers to 'complete capture histories'. The other options are explained in detail in the 'big blue book', where **RELEASE** is first described. Most of the time, you're likely to be using CCH, so

this option is selected by default.

Now you're ready to start the simulations. If you haven't completed all of the 'tabbed windows', the 'start simulations button' will be greyed-out (see bottom of previous page). If you are ready to run the simulations, this button will be active. Go ahead and run the simulations. If all goes well, the first thing you'll see is a Notepad window showing the basic **RELEASE** output - this can be ignored for the moment.

In addition (and most important for our purposes in this case), a file called `GOFFILE.dat` will be created on your computer. This file contains all of the results of the individual χ^2 tests, and **TEST 2** and **TEST 3** separately, as well as the sum of **TEST 2** + **TEST 3**. At this point, you'll need to parse this output file to extract just the **TEST 2** + **TEST 3** results. We'll leave details of how to do this up to you. But, for completeness, here are the results of our simulation. If we plot $\hat{c}$ estimated as (**TEST2** + **TEST3**)/df



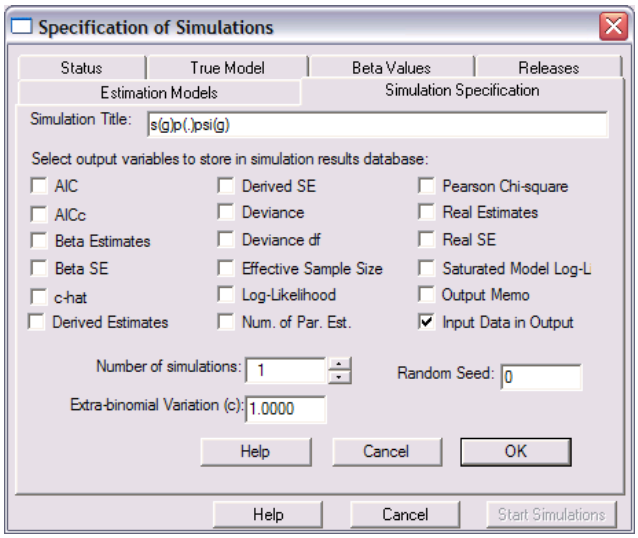
then we see that even though the true value for c in the simulated data is 1.0, there is considerable variation among simulated data sets in estimated $\hat{c}$. The implications of this uncertainty are discussed in some detail in Chapter 5.

A.2. Generating encounter histories - program MARK

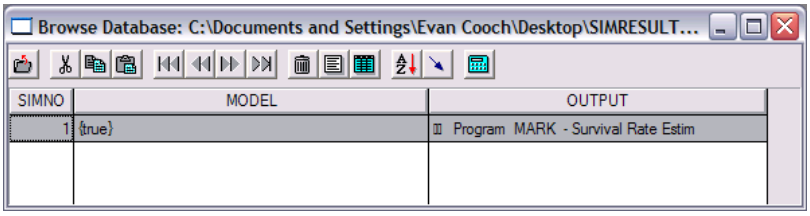
Here, we simulate some multi-state data. Our purpose here is primarily to illustrate some of the other features of the simulation tool in **MARK** - in particular, the ability to extract the encounter histories generated by the simulations (up until now, we've only considered the analysis results, not the encounter histories themselves). We'll simulate $\{S_g p \psi_g\}$ - simple group differences in survival and movement (no time variation), but no differences in encounter probability between groups, or over time.

As with earlier examples, start **MARK**, and and start the setup for a new **MARK** simulation. Select '**`Multi-strata recaptures`**' as the data type, 6 occasions, and 2 strata. Then, using the PIM chart, setup $\{S_g p \psi_g\}$ as the true model. Use whatever values you like for each of the parameters, and 100 releases of newly marked individuals on each occasion. Since we're interested in looking at the encounter histories for the true model, we use the current PIM structure for the '**`Estimation Model`**'.

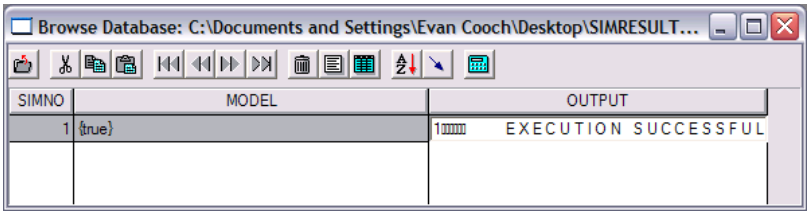
Finally, the only thing we need to pay any real attention to is the '**Simulation Specification**' tab. Here, we want to specify only a single simulation (i.e., change the '**Number of simulations**' to 1). Then, we check '**Input Data in Output**' - this will cause the input data (i.e., the simulated encounter histories) to be written into the simulation output.



Because you're only running 1 simulation, it will run very quickly. Next step is to '**View Simulation Results**'. We'll start with the standard '**Browser View**'.



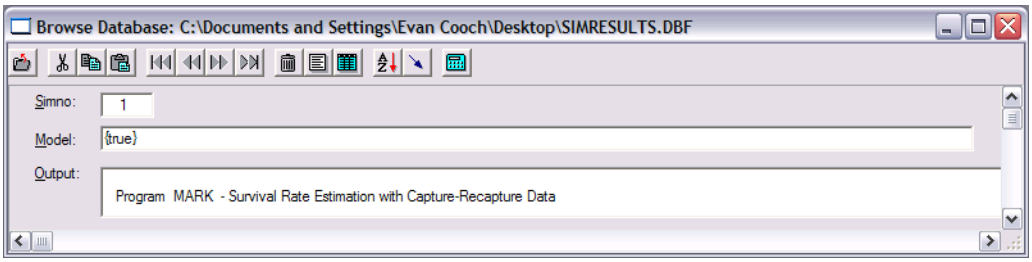
We see that the browser contains the simulation number (only 1, in this case, see we ran only a single simulation), the name of the model, and then a column labeled '**Output**'. To extract the encounter histories, double-click the cell in the '**Output**' column. When you do so, you'll see that the text in the cell changes:



What you're actually seeing is the first line of the entire (full) output that **MARK** generates. This full output includes the encounter histories. How do you actually access these histories (given that you

can't scroll down the file from within the browser)? What you need to do is copy the contents of the cell - you can do this either by using `ctrl-A`, or by right-clicking from within the cell, and selecting '**Copy**'. Once you have copied the cell contents into the clipboard, you next need to paste the contents into your favorite editor. After doing so, scroll down a few lines until you find the encounter histories. All that remains is to extract the encounter histories from this file (this is either easy or difficult, depending on your choice of editor). Thats it!

Earlier, we mentioned that there was another '**view**' option for looking at the results of your simulations. You can also look at your results using the '**Form**' view option. To access the '**Form**' view, simply view the simulation results. **MARK** will typically default to the browser view. You can switch to the form view either by selecting the appropriate button on the toolbar, or using the appropriate view menu option. For the current simulation of multi-state data, here is what the form view would look like:



As with the browser option, you can extract the encounter histories simply by selecting the text in the output window, and then copying the text to your favorite editor.

A.3. simulating data from a prior MARK analysis

Occasionally, it will be of interest to simulate data based on model structures you may have constructed for some other **MARK** analysis. How can you 'pull in' the model structure from another **MARK** project file into the simulation tool in **MARK**?

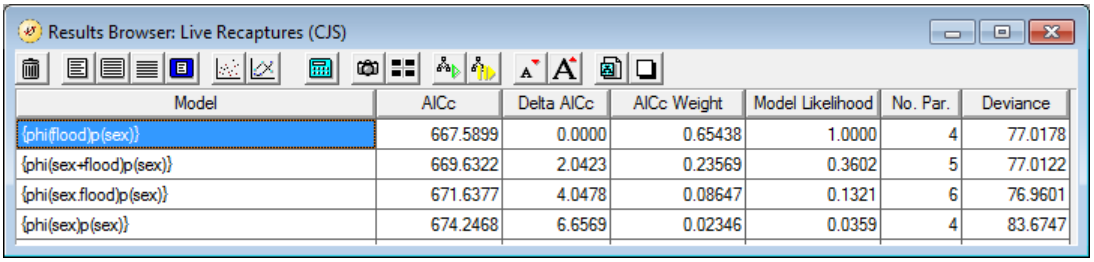
In fact, it isn't really that hard. There are at least a couple of ways you can do this. In one approach, you simply

1. open up the **MARK** project of interest
2. retrieve the model of interest
3. then, from within the open **MARK** project, select '**Simulations | Setup Simulations**', and then '**override**' the popup window.
4. this will spawn the 'specification of simulations' window introduced earlier. Simply click the '**true model**' tab, and then click the '**current PIM model**' button. This will make the 'active model' (which you retrieved in step 2, above) the 'active model' for the simulation.
5. remaining steps are pretty much as described earlier. However, in this case, the number of releases is 'set' to equal the number of releases in the **MARK** analysis. You can change this, if you like. The default link function is left at '**identity**', and the default β parameters are left at the default value of 0.5.

Alternatively, you can

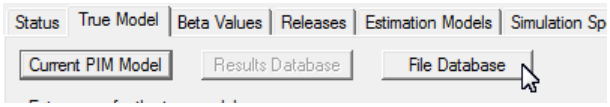
1. open up **MARK**, and select '**Simulations | Setup Simulations**', and then '**override**' the popup window.
2. enter the details for the analysis you're interested in simulating (e.g., if you want to run one or more models from an analysis you did of a data set consisting of live encounters, 7 encounter occasions, an 2 groups, then you need to enter this data specification).
3. Once you've finished step (3), and clicked '**OK**', this will spawn the 'specification of simulations' window introduced earlier. Simply click the '**true model**' tab.
4. Look for a button labeled '**File database**', and click it. This will let you browse to a particular `.dbf` file for some other **MARK** project, select it, and then pick the model you want to simulate from those contained in the `.dbf` file (note, you can only pick one model at a time)
5. remaining steps are pretty much as described earlier. However, in this case, the link function, the beta parameters (for the link function), but not the number of releases is 'set' to equal the values from the **MARK** analysis. Again, you can manually override whatever you like.

We'll demonstrate the second approach using the full Dipper data set. Recall that the Dipper data consists of live encounter data, 7 occasions, and 2 groups (males and females). We'll assume we've already analyzed these data (say the results are contained in `ED.DBF`), fitting 4 different models. The 4 models are shown in the following browser:



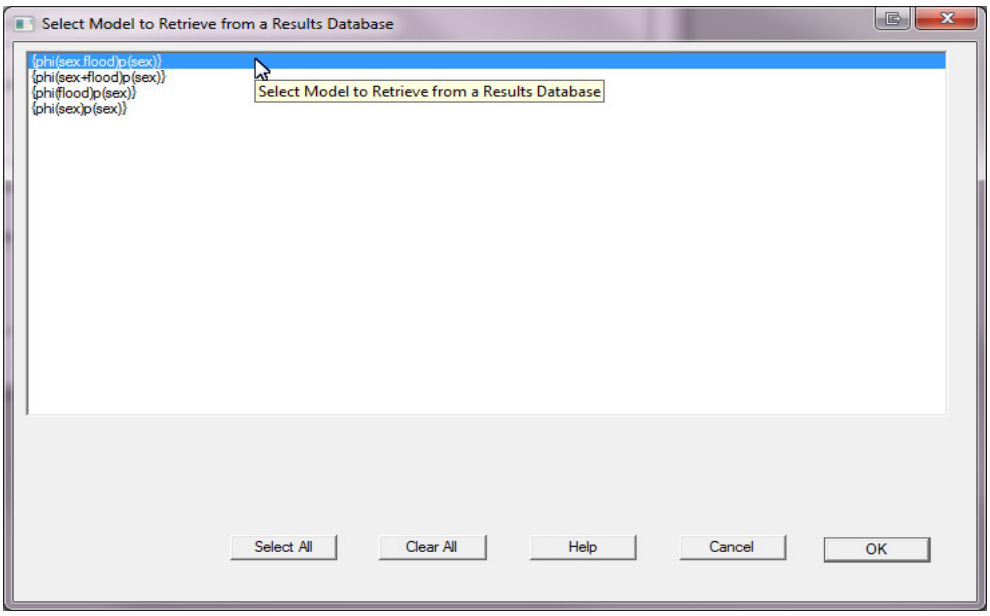
Model	AICc	Delta AICc	AICc Weight	Model Likelihood	No. Par.	Deviance
{phi(flood)p(sex)}	667.5899	0.0000	0.65438	1.0000	4	77.0178
{phi(sex+flood)p(sex)}	669.6322	2.0423	0.23569	0.3602	5	77.0122
{phi(sex.flood)p(sex)}	671.6377	4.0478	0.08647	0.1321	6	76.9601
{phi(sex)p(sex)}	674.2468	6.6569	0.02346	0.0359	4	83.6747

Suppose we want to simulate data under the model $\{\varphi_{sex.flood}p_{sex}\}$. From above, all we need to do is start **MARK**, and select '**Simulations | Setup Simulations**', and then '**override**' the popup window. Next, we click the appropriate data type ('**Recaptures only**'), and specify 7 occasions, and 2 groups (it isn't necessary to label the groups, or to make the group labels equivalent to the labels used in the original Dipper analysis). Then, in the '**Specification of Simulations**' window, select the '**True Model**' tab. Click the '**File Database**' button



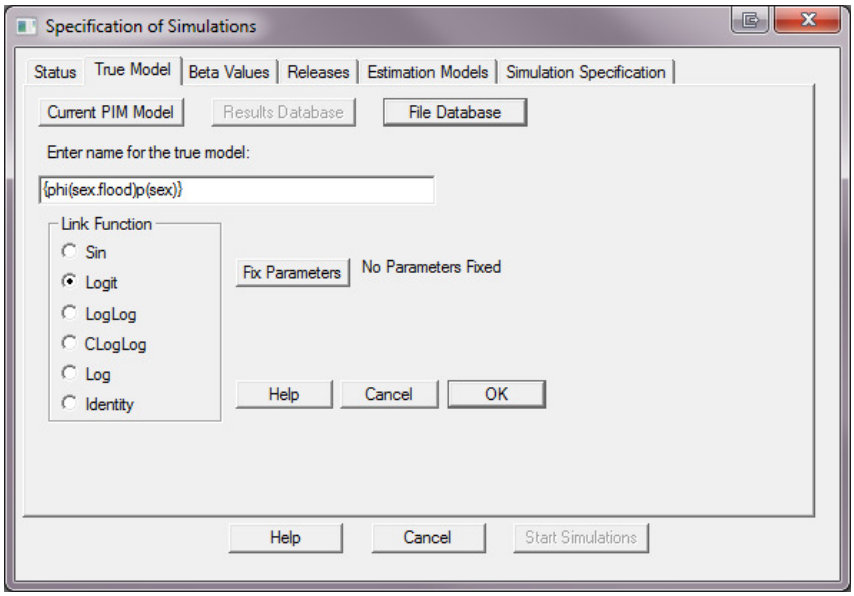
and browser to '`ED.DBF`' (i.e., the `.dbf` file containing the results of your earlier Dipper analysis).

Once you've selected `ED.DBF`, you will be presented with a list of all 4 of the models contained in the results database for the Dipper analysis. Simply highlight/select the model you want to simulate from the list (as shown below, for model $\{\varphi_{sex.flood}p_{sex}\}$):

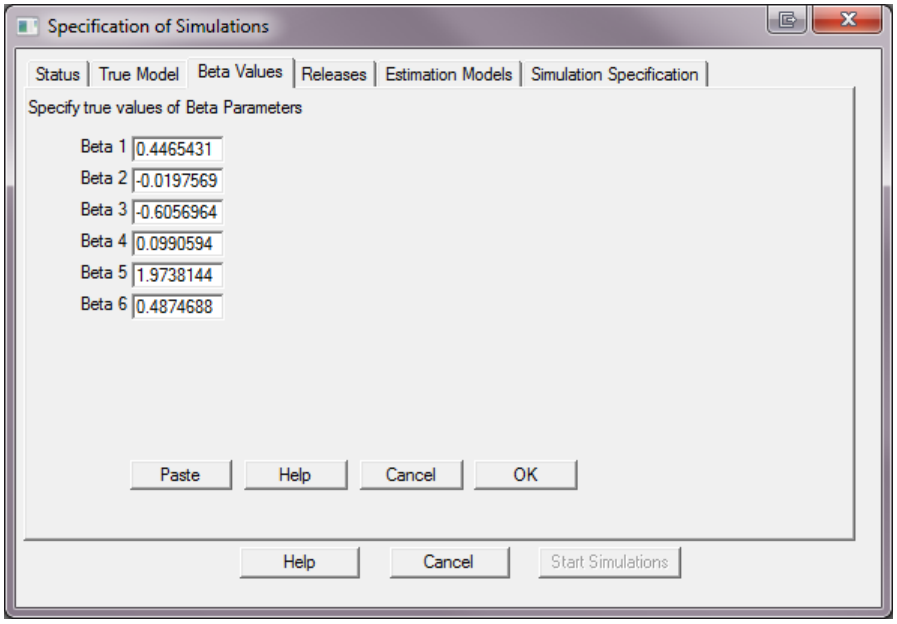


Note: don't click the '**`Select all`**' option. Trying to bring all the files models into the **MARK** simulation tool at the same time will cause **MARK** to crash. Retrieve one file model at a time.

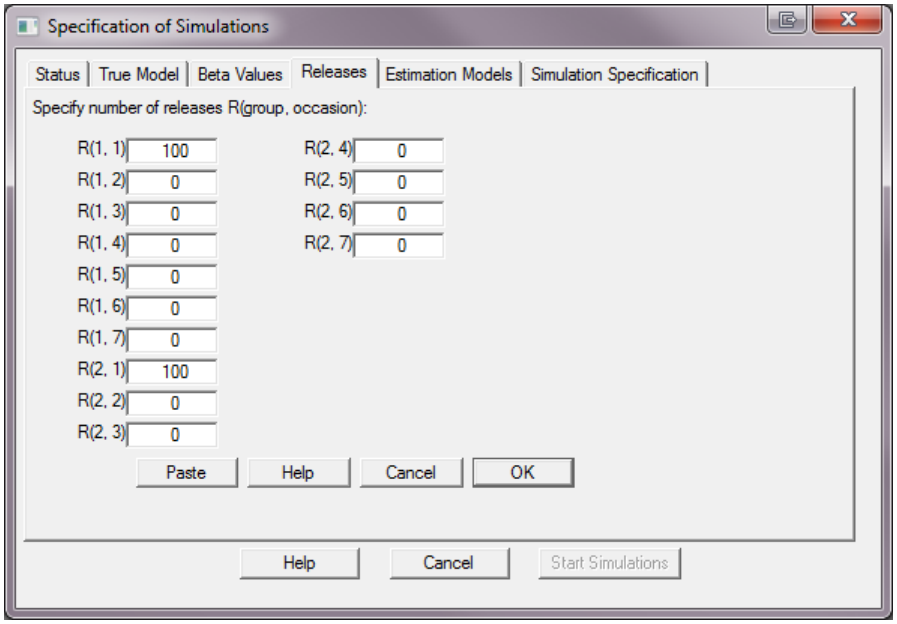
Once you retrieve the file model, **MARK** will drop you back into the '**`Specification of Simulations`**' window. You'll see (below) that the name for the true model has been set (reflecting the name of the file model you just retrieved). It is also important to note that **MARK** has also set the link function to whatever link function you used for that model in `ED.DBF` (in this case, the logit link).



Click the '**OK**' button. Next, if you click the '**Beta values**' tab, you'll see that **MARK** has also 'retrieved' the beta value estimates from the file model:



However, as noted earlier, **MARK** does not retrieve the original number of releases from the file model:



In summary, it is relatively easy to 'retrieve' a previous file structure into the simulation capability of **MARK**. Simply use a method which is most convenient for your purposes.

A.4. Simulation of robust design + closed capture data - special considerations

Simulation of the robust design model (Chapter 15) requires that the user enter parameters for the simulation that are interpreted differently than what is estimated. This issue occurs for the population estimates for the second and later primary sessions. The user would expect that the values entered for the population sizes for each primary session would be the actual population sizes. This assumption is true for the first primary session. However, for the second and later primary sessions, the value of N entered is the number of new animals entering the population at that point, i.e., N is now the number of animals available for capture *that have never previously been available for capture*. This rather strange arrangement is caused because of the way that the robust design models are structured (see Chapter 15).

For the closed captures models (see Chapter 14) in the robust design where N is in the likelihood, the values entered for the N parameters are always assumed to have the identity link. Thus, regardless of what link function is specified for the true model, the values entered for the N parameters are the values for N. For the Huggins closed captures models (where N does not appear in the likelihood), N values are entered in a separate tab window labeled as the true population size. However, again, the actual values entered are either the initial population size (first primary session) or else the number of new animals entering the population (second and later primary sessions).

A.5. Summary

The simulation capability of **MARK** is a very powerful and effective way to look at the impacts of number of occasions, sample size, number of releases, and other factors, on the strength of your inference, before you actually conduct your study. Conducting a study involving marked individuals requires careful planning, and the simulation tool in **MARK** is an effective first step.

APPENDIX B

The 'Delta method' . . .

Suppose you have conducted a mark-recapture study over 4 years which yields 3 estimates of apparent annual survival (say, $\hat{\varphi}_1$, $\hat{\varphi}_2$, and $\hat{\varphi}_3$). But, suppose what you are really interested in is the estimate of the product of the three survival values (i.e., the probability of surviving from the beginning of the study to the end of the study)? While it is easy enough to derive an estimate of this product (as $[\hat{\varphi}_1 \times \hat{\varphi}_2 \times \hat{\varphi}_3]$), how do you derive an estimate of the *variance* of the product? In other words, how do you derive an estimate of the variance of a transformation of one or more random variables (in this case, we transform the three random variables - $\hat{\varphi}_i$ - by considering their product)?

One commonly used approach which is easily implemented, not computer-intensive, and can be robustly applied in many (but not all) situations is the so-called *Delta method* (also known as the method of propagation of errors). In this appendix, we briefly introduce the underlying background theory, and the application of the Delta method, to fairly typical scenarios.

B.1. Mean and variance of random variables

Our interest here is developing a method that will allow us to estimate the variance for functions of random variables. Let's start by considering the formal approach for deriving these values explicitly, based on the *method of moments*. For continuous random variables, consider a continuous function $f(x)$ on the interval $[-\infty, +\infty]$. The first three moments of $f(x)$ can be written as:

$$M_0 = \int_{-\infty}^{+\infty} f(x)dx$$

$$M_1 = \int_{-\infty}^{+\infty} x f(x)dx$$

$$M_2 = \int_{-\infty}^{+\infty} x^2 f(x)dx$$

In the particular case that the function is a probability density (as for a continuous random variable), then $M_0 = 1$ (i.e., the area under the pdf must equal 1).

For example, consider the uniform distribution on the finite interval $[a, b]$. A uniform distribution (sometimes also known as a rectangular distribution), is a distribution that has constant probability over the interval. The probability density function (pdf) for a continuous uniform distribution on the

finite interval $[a, b]$ is:

$$P(x) = \begin{cases} 0 \text{ for } x < a \\ 1/(b-a) \text{ for } a < x < b \\ 0 \text{ for } x > b \end{cases}$$

Integrating the pdf for $p(x) = 1/(b-a)$:

$$\begin{aligned} M_0 &= \int_a^b p(x)dx \\ &= \int_a^b \frac{1}{b-a}dx = 1 \end{aligned} \tag{B.1}$$

$$\begin{aligned} M_1 &= \int_a^b xp(x)dx \\ &= \int_a^b \frac{x}{b-a}dx = \frac{a+b}{2} \end{aligned} \tag{B.2}$$

$$\begin{aligned} M_2 &= \int_a^b x^2 p(x)dx \\ &= \int_a^b x^2 \frac{1}{b-a}dx = \frac{1}{3}\left(a^2 + ab + b^2\right) \end{aligned} \tag{B.3}$$

We see clearly that M_1 is the mean of the distribution. What about the variance? Where does the second moment M_2 come in? Recall that the variance is defined as the average value of the fundamental quantity [`distance from mean`]^2. The squaring of the distance is so the values to either side of the mean don't cancel out. Standard deviation is simply the square-root of the variance.

Given some discrete random variable x_i, with probability p_i, and mean μ, we define the variance as:

$$\text{var} = \sum (x_i - \mu)^2 p_i$$

Note we don't have to divide by the number of values of x because the sum of the discrete probability distribution is 1 (i.e., $\sum p_i = 1$). For a continuous probability distribution, with mean μ, we define the variance as:

$$\text{var} = \int_a^b (x-\mu)^2 p(x)dx$$

Given our moment equations, we can then write:

$$\begin{aligned} \text{var} &= \int_a^b (x-\mu)^2 p(x)dx \\ &= \int_a^b \left(x^2 - 2\mu x + \mu^2\right)p(x)dx \\ &= \int_a^b x^2 p(x)dx - \int_a^b 2\mu x p(x)dx + \int_a^b \mu^2 p(x)dx \\ &= \int_a^b x^2 p(x)dx - 2\mu \int_a^b xp(x)dx + \mu^2 \int_a^b p(x)dx \end{aligned}$$

Now, if we look closely at the last line, we see that in fact the terms represent the different moments of the distribution. Thus we can write:

$$\begin{aligned}\text{var} &= \int_a^b (x-\mu)^2 p(x)dx \\ &= \int_a^b x^2 p(x)dx - 2\mu\int_a^b xp(x)dx + \mu^2\int_a^b p(x)dx \\ &= M_2 - 2\mu(M_1) + \mu^2(M_0)\end{aligned}$$

Since $M_1 = \mu$, and $M_0 = 1$ then:

$$\begin{aligned}\text{var} &= M_2 - 2\mu(M_1) + \mu^2(M_0) \\ &= M_2 - 2\mu(\mu) + \mu^2(1) \\ &= M_2 - 2\mu^2 + \mu^2 \\ &= M_2 - \mu^2 \\ &= M_2 - (M_1)^2\end{aligned}$$

In other words, the variance for the pdf is simply the second moment (M_2) minus the square of the first moment ($(M_1)^2$). Thus, for a continuous uniform random variable x on the interval $[a, b]$:

$$\begin{aligned}\text{var} &= M_2 - (M_1)^2 \\ &= \frac{(a-b)^2}{12}\end{aligned}$$

B.2. Transformations of random variables and the Delta method

OK - that's fine. If the pdf is specified, we can use the method of moments to formally derive the mean and variance of the distribution. But, what about functions of random variables having poorly specified or unspecified distributions? Or, situations where the pdf is not easily defined?

In such cases, we may need other approaches. We'll introduce one such approach (the Delta method) here, by considering the case of a simple linear transformation of a random normal distribution.

Let

$$X_1, X_2, \ldots \sim N(10, \sigma^2 = 2)$$

In other words, random deviates drawn from a normal distribution with a mean of 10, and a variance of 2. Consider some transformations of these random values. You might recall from some earlier statistics or probability class that linearly transformed normal random variables are themselves normally distributed. Consider for example, $X_i \sim N(10, 2)$ - which we then linearly transform to Y_i, such that $Y_i = 4X_i + 3$.

Now, recall that for real scalar constants a and b we can show that

i. $E(a) = a$, $E(aX + b) = aE(X) + b$
ii. $\text{var}(a) = 0$, $\text{var}(aX + b) = a^2\text{var}(X)$

Thus, given $X_i \sim N(10, 2)$ and the linear transformation $Y_i = 4X_i + 3$, we can write:

$$Y \sim N\Big([4(10) + 3 = 43], [(4^2)(2)]\Big) = N(43, 32)$$

Now, an important point to note is that some transformations of the normal distribution are close to normal (i.e., are linear) and some are not. Since linear transformations of random normal values are normal, it seems reasonable to conclude that approximately linear transformations (over some range) of random normal data should also be approximately normal.

OK, to continue. Let $X \sim N(\mu, \sigma^2)$,and let $Y = g(X)$, where g is some transformation of X (in the previous example, $g(X) = 4X+3$). It is hopefully relatively intuitive that the closer $g(X)$ is to linear over the likely range of X (i.e., within 3 or so standard deviations of μ), the closer $Y = g(X)$ will be to normally distributed. From calculus, we recall that if you look at any differentiable function over a narrow enough region, the function appears approximately linear. The approximating line is the tangent line to the curve, and its slope is the derivative of the function.

Since most of the mass (i.e., most of the random values) of X is concentrated around μ, let's figure out the tangent line at μ, using two different methods. First, we know that the tangent line passes through $(\mu, g(\mu))$, and that its slope is $g'\mu$ (we use the 'prime' notation, g', to indicate the first derivative of the function g). Thus, the equation of the tangent line is $Y = g'X + b$ for some b. Replacing (X, Y) with the known point $(\mu, g(\mu))$, we find $g(\mu) = g'(\mu)\mu + b$ and so $b = g(\mu) - g'(\mu)\mu$. Thus, the equation of the tangent line is $Y = g'(\mu)X + g(\mu) - g'(\mu)\mu$.

Now for the big step – we can derive an approximation to the same tangent line by using a *Taylor series expansion* of $g(x)$ (to first order) around $X = \mu$:

$$\begin{aligned} Y &= g(X) \\ &\approx g(\mu) + g'(\mu)(X - u) + \epsilon \\ &= g'(\mu)X + g(\mu) - g'(\mu)\mu + \epsilon \end{aligned}$$

OK, at this point you might be asking yourself 'so what?'. Well, suppose that $X \sim N(\mu, \sigma^2)$ and $Y = g(X)$, where $g'(\mu) \neq 0$. Then, whenever the tangent line (derived earlier) is approximately correct over the likely range of X (i.e., if the transformed function is approximately linear over the likely range of X), then the transformation $Y = g(X)$ will have an approximate normal distribution. That approximate normal distribution may be found using the usual rules for linear transformations of normals.

Thus, to first order:

$$E(Y) \approx g'(\mu)\mu + g(\mu) - g'(\mu)\mu = g(\mu)$$

$$\begin{aligned} \text{var}(Y) \approx \text{var}(g(X)) &= \Big(g(X) - g(\mu)\Big)^2 \\ &= (g'(\mu)(X - \mu))^2 \\ &= (g'(\mu))^2(X - \mu)^2 \\ &= (g'(\mu))^2\text{var}(X) \end{aligned}$$

In other words, we take the derivative of the transformed function with respect to the parameter, square it, and multiply it by the estimated variance of the untransformed parameter.

These first-order approximations to the variance of a transformed parameter are usually referred to as the *Delta method*.*

* For an interesting review of the history of the Delta method, see Ver Hoef, Jay M. (2012) Who Invented the Delta Method? *The American Statistician*, **66**, 124-127.

begin sidebar

Taylor series expansions?

A very important, and frequently used tool. If you have no familiarity at all with series expansions, here is a (very) short introduction. Briefly, the *Taylor series* is a power series expansion of an infinitely differentiable real (or complex) function defined on an open interval around some specified point. For example, a one-dimensional Taylor series is an expansion of a real function $f(x)$ about a point $x = a$ over the interval $(a - r, a + r)$, is given as:

$$f(x) \approx f(a) + \frac{f'(a)(x-a)}{1!} + \frac{f''(x)(x-a)^2}{2!} + \ldots$$

where $f'(a)$ is the first derivative of f with respect to a, $f''(x)$ is the second derivative of f with respect to a, and so on.

For example, suppose the function is $f(x) = e^x$. The convenient fact about this function is that all its derivatives are equal to e^x as well (i.e., $f(x) = e^x$, $f'(x) = e^x$, $f'' = e^x, \ldots$). In particular, $f^{(n)}(x) = e^x$ so that $f^{(n)}(0) = 1$. This means that the coefficients of the Taylor series are given by:

$$a_n = \frac{f^{(n)}(0)}{n!} = \frac{1}{n!},$$

and so the Taylor series is given by:

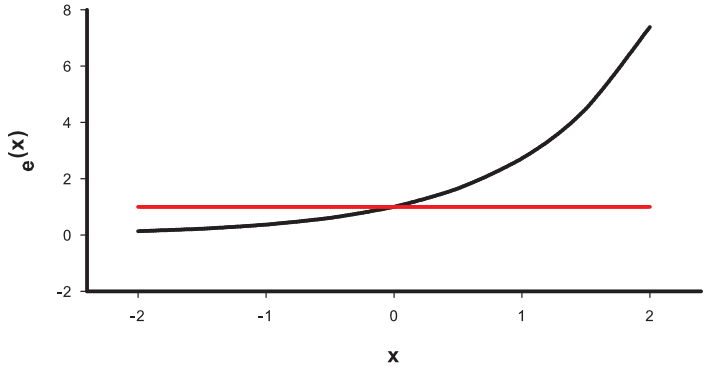
$$1 + x + \frac{x^2}{2} + \frac{x^3}{6} + \frac{x^4}{24} + \ldots + \frac{x^n}{n!} + \ldots = \sum_{n=0}^{\infty} \frac{x^n}{n!}$$

The primary utility of such a power series in simple application is that differentiation and integration of power series can be performed term by term and is hence particularly (or, at least relatively) easy. In addition, the (truncated) series can be used to compute function values approximately.

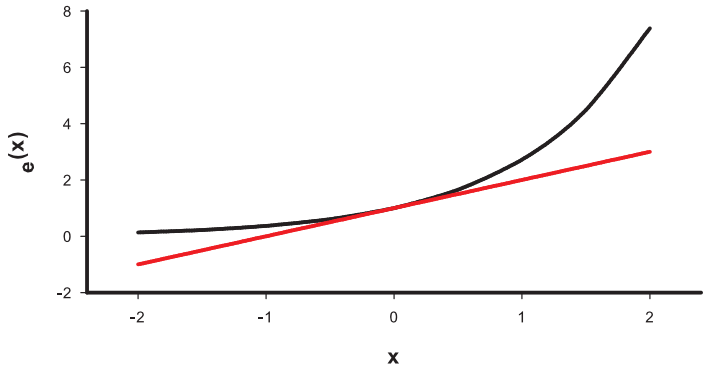
Now, let's look at an example of the "fit" of a Taylor series to a familiar function, given a certain number of terms in the series. For our example, we'll expand the function $f(x) = e^x$, at $x = a = 0$, on the interval $[a - 2, a + 2]$, for $n = 0$, $n = 1$, $n = 2, \ldots$ (where n is the number of terms in the series). For $n = 0$, the Taylor expansion is a scalar constant (1):

$$f(x) \approx 1$$

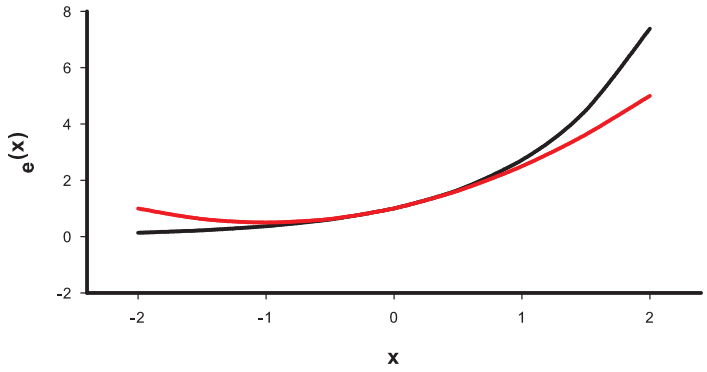
which is obviously a poor approximation to the function $f(x) = e^x$ at any point. This is shown clearly in the following figure - the black line in the figure is the function $f(x) = e^x$, evaluated over the interval $[-2, 2]$, and the red line is the Taylor series approximation for $n = a = 0$.



What happens when we add higher order terms? Here is the plot of the Taylor series for $n = 1$:



Hmmm...a bit better. What about $n = 2$?



We see that when we add more terms (i.e., use a higher-order series), the fit gets progressively better. Often, for 'nice, smooth' functions (i.e., those nearly linear at the point of interest), we don't need many terms at all. For this example, $n = 4$ yields a near-perfect fit (over the interval $[-2, 2]$.

Another example - suppose the function of interest is $f(x) = (x)^{1/3}$ (i.e., $f(x) = \sqrt[3]{x}$). Suppose we're interested in $f(x) = (x)^{1/3}$ where $x = 27$ (i.e., $f(27) = \sqrt[3]{27}$). Now, it is straightforward to show that $f(27) = \sqrt[3]{27} = 3$. But suppose we want to know $f(25) = \sqrt[3]{25}$, using a Taylor series approximation? We recall that to first order:

$$f(x) = f(a) + f'(a)(x - a),$$

where in this case, $a = 25$ and $x = 27$. The derivative of f with respect to x for this function $f(a) = (a)^{1/3}$ is:

$$f'(a) = \frac{a^{-2/3}}{3} = \frac{1}{3\sqrt[3]{x^2}}$$

Thus, using the first-order Taylor series, we write:

$$\begin{aligned} f(25) &\approx f(27) + f'(27)(25 - 27) \\ &= 3 + (0.037037)(-2) \\ &= 3 - 0.0740741 \end{aligned}$$

$$= 2.926$$

Clearly, 2.926 is very close to the true value of $f(25) = \sqrt[3]{25} = 2.924$. In other words, the first-order Taylor approximation works well for this function. As we will see later, this is not always the case.

end sidebar

B.3. Transformations of one variable

OK, enough background for now. Let's see some applications. Let's check the Delta method out in a case where we know the answer. Assume we have an estimate of density $\hat{D}$ and its conditional sampling variance, $\widehat{\text{var}}(\hat{D})$. We want to multiply this by some constant c to make it comparable with other values from the literature. Thus, we want $\hat{D}_s = g(D) = c\hat{D}$, and $\widehat{\text{var}}(D_s)$.

The Delta method gives:

$$\begin{aligned}\widehat{\text{var}}(\hat{D}_s) &\approx (g'(D))^2 \dot{\sigma}_D^2 \\ &= \left(\frac{\partial \hat{D}_s}{\partial \hat{D}}\right)^2 \cdot \widehat{\text{var}}(\hat{D}) \\ &= c^2 \cdot \widehat{\text{var}}(\hat{D})\end{aligned}$$

which we know to be true for the variance of a random variable multiplied by a real constant.

Another example of the same thing – consider a known number of harvested fish and an average weight ($\hat{\mu}_w$) and its variance. If you want an estimate of total biomass (B), then $\hat{B} = N \cdot \hat{\mu}_w$ and the variance of $\hat{B}$ is $N^2 \cdot \widehat{\text{var}}(\mu_w)$.

Still another example – you have some parameter θ, which you transform by dividing it by some constant c. Thus, by the Delta method:

$$\widehat{\text{var}}\left(\frac{\hat{\theta}}{c}\right) \approx \left(\frac{1}{c}\right)^2 \cdot \widehat{\text{var}}(\hat{\theta})$$

B.3.1. A potential complication - violation of assumptions

A final – and conceptually important – example for transformations of single variables. The importance lies in the demonstration that the Delta method does not always work. Remember, the Delta method assumes that the transformation is approximately linear over the expected range of the parameter. Suppose one has an MLE for the mean and estimated variance for some parameter θ which is bounded random uniform on the interval [0, 2]. Suppose you want to transform this parameter such that:

$$\psi = e^{\theta}$$

[Recall that this is a convenient transformation since the derivative of e^x is e^x, making the calculations very simple].

Now, based on the Delta method, the variance for ψ would be estimated as:

$$\widehat{\text{var}}(\hat{\psi}) \approx \left(\frac{\partial \hat{\psi}}{\partial \hat{\theta}}\right)^2 \cdot \widehat{\text{var}}(\hat{\theta})$$

$$= \left(e^{\theta}\right)^2 \cdot \widehat{\text{var}}(\hat{\theta})$$

Now, suppose that $\hat{\theta} = 1.0$, and $\widehat{\text{var}}(\hat{\theta}) = 0.3\dot{3}$. Then, from the Delta method:

$$\begin{aligned}\widehat{\text{var}}(\hat{\psi}) &\approx \left(e^{\theta}\right)^2 \cdot \widehat{\text{var}}(\hat{\theta}) \\ &= (7.38906)(0.3\dot{3}) \\ &= 2.46302\end{aligned}$$

OK, so what's the problem? Well, let's derive the variance of ψ using the method of moments. To do this, we need to integrate the pdf (uniform, in this case) over some range. Since the variance of a uniform distribution is $(b-a)^2/12$, and if b and a are symmetric around the mean (1.0), then we can show by algebra that given a variance of $0.3\dot{3}$, then $a = 0$ and $b = 2$.

Given a uniform distribution, the pdf is $p(\theta) = 1/(b-a)$. Thus, by the method of moments:

$$\begin{aligned}M_1 &= \int_a^b g(x)p(x)dx \\ &= \int_a^b \frac{g(x)}{b-a}dx \\ &= -\frac{e^b - e^a}{a-b}\end{aligned}$$

$$\begin{aligned}M_2 &= \int_a^b \frac{g(x)^2}{b-a}dx \\ &= \left(\frac{1}{2}\right) \cdot \frac{e^{2a} - e^{2b}}{a-b}\end{aligned}$$

Thus, by moments, $\text{var}(E(\psi)$ is:

$$\text{var}(E(\psi)) = M_2 - (M_1)^2 = \left(\frac{1}{2}\right) \cdot \frac{-e^{2b} + e^{2a}}{-b+a} - \frac{(e^b - e^a)^2}{(a-b)^2}$$

If $a = 0$ and $b = 2$, then:

$$\begin{aligned}\text{var}(E(\psi)) &= M_2 - (M_1)^2 \\ &= \left(\frac{1}{2}\right) \cdot \frac{-e^{2b} + e^{2a}}{-b+a} - \frac{(e^b - e^a)^2}{(a-b)^2} \\ &= 3.19453,\end{aligned}$$

which is not particularly close to the value estimated by the Delta method (2.46302).

Why the discrepancy? As discussed earlier, the Delta method rests on the assumption the first-order Taylor expansion around the parameter value is effectively linear over the range of values likely to be encountered. Since in this example we're using a uniform pdf, then all values between a and b are equally likely. Thus, we might anticipate that as the interval between a and b gets smaller, then the approximation to the variance (which will clearly decrease) will get better and better (since the smaller the interval, the more likely it is that the function is approximately linear over that range). For example, if $a = 0.5$ and $b = 1.5$ (same mean of 1.0), then the true variance of θ will be $0.08\dot{3}$. Thus, by the

Delta method, the estimated variance of ψ will be 0.61575, while by the method of moments (which is exact), the variance will be 0.65792. Clearly, the proportional difference between the two values has declined markedly. But, we achieved this 'improvement' by artificially reducing the true variance of the untransformed variable θ. Obviously, we can't do this in general practice.

So, what are the practical options? Well, one possible solution is to use a higher-order Taylor series approximation - by including higher-order terms, we can achieve a better 'fit' to the function (see the preceding `sidebar`). If we used a second-order TSE:

$$E(g(x)) \approx g(\mu) + \frac{1}{2}g''(\mu)\sigma^2 \tag{B.4}$$

$$\text{var}(g(x)) \approx g'(\mu)^2\sigma^2 + \frac{1}{4}(g''(\mu))^2(\text{var}(x^2) - 4\mu^2\sigma^2) \tag{B.5}$$

we should do a bit better. For the variance estimate, we need to know $\text{var}(x^2)$, which for a continuous uniform distribution by the method of moments is:

$$\left(\frac{b^5 - a^5}{b - a}\right) - \left(\frac{(b^3 - a^3)^2}{(b - a)^2}\right)$$

Thus, from the second-order approximation, and again assuming $a = 0$ and $b = 2$, then $\widehat{\text{var}}(\hat{\psi})$ is:

$$\begin{aligned}\widehat{\text{var}}(\hat{\psi}) &\approx \left(e^{\hat{\theta}}\right)^2 \cdot \widehat{\text{var}}(\hat{\theta}) + \frac{1}{4}\left(e^{\hat{\theta}}\right)^2 \cdot \widehat{\text{var}}(\hat{\theta}^2) - 4\mu^2\widehat{\text{var}}(\hat{\theta}) \\ &= 3.756316,\end{aligned}$$

which is closer (proportionately) to the true variance (3.19453) than was the estimate using only the first-order TSE (2.46302). The reason that even a second-order approximation isn't 'much closer' is because the transformation is very non-linear over the range of the data (uniform $[0, 2]$ in this case), such that the second-order approximation doesn't 'fit' particularly well over this range.

So, we see that the classical Delta method, which is based on a first-order Taylor series expansion of the transformed function, may not do particularly well if the function is highly non-linear over the range of values being examined. Of course, it would be fair to note that the preceding example made the assumption that the distribution was random uniform over the interval. For most of our work with **MARK**, the interval is likely to have a symmetric mass around the estimate, typically β. As such, most of data, and thus the transformed data, will actually fall closer to the parameter value in question (the mean in this example) than we've demonstrated here. So much so, that the discrepancy between the first order 'Delta' approximation to the variance and the true value of the variance will likely be significantly smaller than shown here, even for a strongly non-linear transformation. We leave it to you as an exercise to prove this for yourself.

But, this point notwithstanding, it is important to be aware of the assumptions underlying the Delta method – if your transformation is non-linear, and there is considerable variation in your data, the first-order approximation may not be particular good. Fortunately, use of second order Taylor series approximations is not heroically difficult – the challenge is usually coming up with $\text{var}(\theta^2)$. If the pdf for the untransformed data is specified (which is essentially equivalent to assuming an informative prior), then you can derive $\text{var}(\theta^2)$ fairly easily using the method of moments.

B.4. Transformations of two or more variables

We are often interested in transformations involving more than one variable. Fortunately, there are also multivariate generalizations of the Delta method.

Suppose you've estimated p different random variables $X_1, X_2, \ldots, X_p$. In matrix notation, these variables would constitute a $(p \times 1)$ random vector:

$$\mathbf{X} = \begin{bmatrix} X_1 \\ X_2 \\ \vdots \\ X_p \end{bmatrix},$$

which has a mean vector:

$$\boldsymbol{\mu} = \begin{bmatrix} EX_1 \\ EX_2 \\ \vdots \\ EX_p \end{bmatrix} = \begin{bmatrix} \mu_1 \\ \mu_2 \\ \vdots \\ \mu_p \end{bmatrix},$$

and the $(p \times p)$ variance-covariance matrix is:

$$\operatorname{cov}(X_1, X_2) = \begin{bmatrix} \operatorname{var}(X_1) & \operatorname{cov}(X_1, X_2) & \ldots & \operatorname{cov}(X_1, X_p) \\ \operatorname{cov}(X_2, X_1) & \operatorname{var}(X_2) & \ldots & \operatorname{cov}(X_2, X_p) \\ \vdots & \vdots & \vdots & \vdots \\ \operatorname{cov}(X_p, X_1) & \operatorname{cov}(X_p, X_2) & \ldots & \operatorname{var}(X_p) \end{bmatrix}$$

Note that if the variables are independent, then the off-diagonal elements (i.e., the covariance terms) are all zero.

Then, for a $(k \times p)$ matrix of constants $\mathbf{A} = a_{ij}$, the expectation of a random vector $\mathbf{Y} = \mathbf{AX}$ is given as:

$$\begin{bmatrix} EY_1 \\ EY_2 \\ \vdots \\ EY_p \end{bmatrix} = \mathbf{A}\boldsymbol{\mu},$$

with a variance-covariance matrix:

$$\operatorname{cov}(\mathbf{Y}) = \mathbf{A}\boldsymbol{\Sigma}\mathbf{A}^{\mathrm{T}}$$

Now, using the same logic we first considered for developing the Delta method for a single variable, for each x_i near μ_i, we can write:

$$y = \begin{bmatrix} g_1(x) \\ g_2(x) \\ \vdots \\ g_p(x) \end{bmatrix} \approx \begin{bmatrix} g_1(\mu) \\ g_2(\mu) \\ \vdots \\ g_p(\mu) \end{bmatrix} + \mathbf{D}(x - \mu),$$

where $\mathbf{D}$ is the matrix of partial derivatives of g_i with respect to x_j, evaluated at $(x - \mu)$.

As with the single-variable Delta method, if the variances of the X_i are small (so that with high probability Y is near μ, such that the linear approximation is usually valid), then to first-order we can write:

$$\begin{bmatrix} EY_1 \\ EY_2 \\ \vdots \\ EY_p \end{bmatrix} = \begin{bmatrix} g_1(\mu) \\ g_2(\mu) \\ \vdots \\ g_p(\mu) \end{bmatrix} \qquad \widehat{\text{var}}(\hat{Y}) \approx \mathbf{D}\boldsymbol{\Sigma}\mathbf{D}^{\text{T}}$$

In other words, to approximate the variance of some multi-variable function $\mathbf{Y}$, we (i) take the vector of partial derivatives of the function with respect to each parameter in turn, $\mathbf{D}$, (ii) right-multiply this vector by the variance-covariance matrix, $\boldsymbol{\Sigma}$, and (iii) right-multiply the resulting product by the transpose of the original vector of partial derivatives, $\mathbf{D}^{\text{T}}$.*

Note: interpretation of the variance estimated using the Delta method is dependent on the source of the variance-covariance matrix, Σ, used in the calculations. If Σ is constructed using standard ML estimates of the variances and covariances, then the resulting Delta method estimate for variance is an estimate of the *total* variance, which is the sum of *sampling + biological process* variance. In contrast, if Σ is based on estimated *process* variances and covariances only, then the Delta method estimate for variance is an estimate of the *process* variance. Decomposition of total variance into sampling and process components is covered in detail in Appendix D.

Example (1) - variance of product of survival probabilities

Let's consider the application of the Delta method in estimating sampling variances of a fairly common function - the product of several parameter estimates.

* There are alternative formulations of this expression which may be more convenient to implement in some instances. When the variables $\theta_1, \theta_2 \ldots \theta_k$ (in the function, Y) are independent, then

$$\begin{aligned} \widehat{\text{var}}(\hat{Y}) &\approx \text{var}(f(\theta_1, \theta_2, \ldots \theta_k)) \\ &= \sum_{i=1}^{k} \text{var}(\theta_i) \left(\frac{\partial f}{\partial \theta_i} \right)^2 \end{aligned}$$

where $\partial f / \partial \theta_1$ is the partial derivative of Y with respect to θ_i. When the variables $\theta_1, \theta_2 \ldots \theta_k$ (in the function, Y) are **not** independent, then the covariance structure among the variables must be accounted for:

$$\begin{aligned} \widehat{\text{var}}(\hat{Y}) &\approx \text{var}(f(\theta_1, \theta_2, \ldots \theta_k)) \\ &= \sum_{i=1}^{k} \text{var}(\theta_i) \left(\frac{\partial f}{\partial \theta_i} \right)^2 + 2 \sum_{i=1}^{k} \sum_{j=1}^{k} \text{cov}(\theta_i, \theta_j) \left(\frac{\partial f}{\partial \theta_i} \right) \left(\frac{\partial f}{\partial \theta_j} \right) \end{aligned}$$

From the preceding, we see that:

$$\widehat{\text{var}}(\hat{Y}) \approx \mathbf{D}\boldsymbol{\Sigma}\mathbf{D}^{\mathrm{T}} = \left[\frac{\partial(\hat{Y})}{\partial(\hat{\theta})}\right] \cdot \widehat{\boldsymbol{\Sigma}} \cdot \left[\frac{\partial(\hat{Y})}{\partial(\hat{\theta})}\right]^{\mathrm{T}},$$

where Y is some linear or nonlinear function of the k parameter estimates $\hat{\theta}_1, \hat{\theta}_2, \ldots \hat{\theta}_k$. The first term, $\mathbf{D}$, on the RHS of the variance expression is a row vector containing partial derivatives of Y with respect to each of these k parameters ($\hat{\theta}_1, \hat{\theta}_2, \ldots \hat{\theta}_k$). The right-most term of the RHS of the variance expression, $\mathbf{D}^{\mathrm{T}}$, is simply a transpose of this row vector (i.e., a column vector). The middle-term, $\boldsymbol{\Sigma}$ is simply the estimated variance-covariance matrix for the parameters.

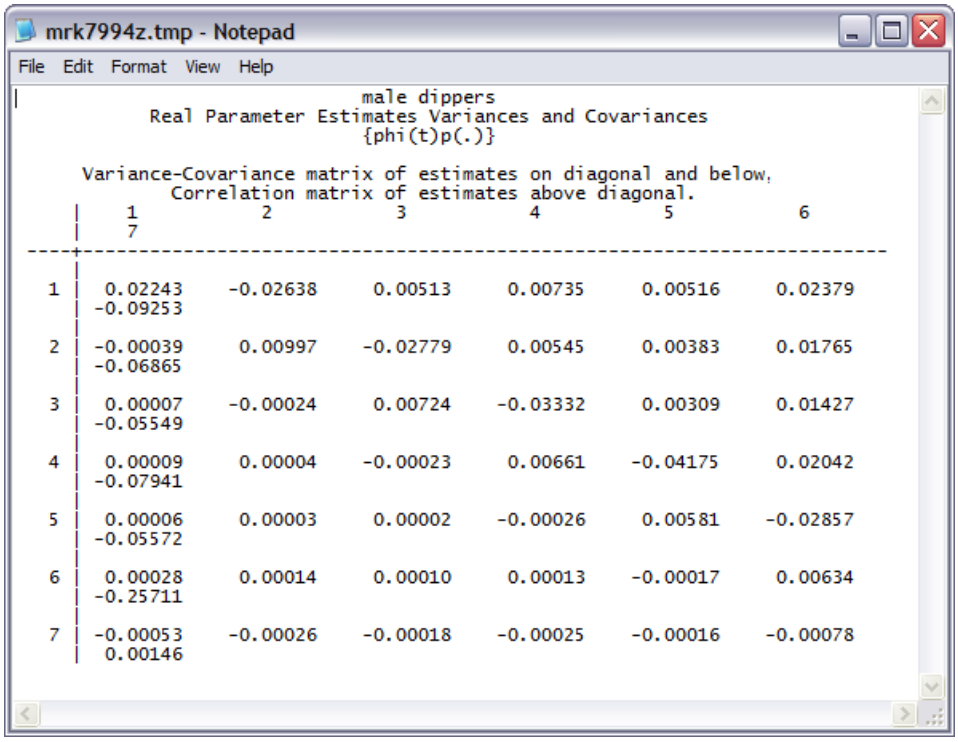
To demonstrate the steps in the calculation, we'll use estimates from the male European dipper data set (yes, again). We'll fit model $\{\varphi_t p_\cdot\}$ to these data. Suppose we're interested in the probability of surviving from the start of the first interval to the end of the third interval. Well, the point-estimate of this probability is easy enough - it's simply

$$\begin{aligned}\hat{Y} &= (\hat{\varphi}_1 \times \hat{\varphi}_2 \times \hat{\varphi}_3) \\ &= (0.6109350 \times 0.458263 \times 0.4960239) = 0.138871\end{aligned}$$

So, the estimated probability of a male Dipper surviving over the first three intervals is ~ 14% (again, assuming that our time-dependent survival model is a valid model).

To derive the estimate of the variance of the product, we will also need the variance-covariance matrix for the survival estimates. You can generate the matrix easily in **MARK** by selecting '**Output | Specific Model Output | Variance-Covariance Matrices | Real Estimates**'.

Here is the variance-covariance matrix for the male Dipper data, generated from model $\{\varphi_t p_\cdot\}$, as output to the default editor (e.g., Windows Notepad):



In the output from **MARK** as shown in the editor, the variance-covariance values are *below* the diagonal, whereas the standardized correlation values are *above* the diagonal. The variances are given *along* the diagonal.

However, it is **very important** to note that the V-C matrix that **MARK** outputs to the editor is *rounded* to 5 significant digits. For the actual calculations, we need to use the full precision values.* To get those, you need to either (i) output the V-C matrix into a dBase file (which you could then open with dBase, or Excel), or (ii) copy the V-C matrix into the Windows clipboard, and then paste it into some other application. Failure to use the full precision V-C matrix will almost always lead to 'rounding errors'.

The 'full precision' V-C matrix for the 3 Dipper survival estimates is shown below.

$$\widehat{\mathrm{cov}}(\hat{Y}) = \widehat{\sum} = \begin{bmatrix} \widehat{\mathrm{var}}(\hat{\varphi}_1) & \widehat{\mathrm{cov}}(\hat{\varphi}_1, \hat{\varphi}_2) & \widehat{\mathrm{cov}}(\hat{\varphi}_1, \hat{\varphi}_3) \\ \widehat{\mathrm{cov}}(\hat{\varphi}_2, \hat{\varphi}_1) & \widehat{\mathrm{var}}(\hat{\varphi}_2) & \widehat{\mathrm{cov}}(\hat{\varphi}_2, \hat{\varphi}_3) \\ \widehat{\mathrm{cov}}(\hat{\varphi}_3, \hat{\varphi}_1) & \widehat{\mathrm{cov}}(\hat{\varphi}_3, \hat{\varphi}_2) & \widehat{\mathrm{var}}(\hat{\varphi}_3) \end{bmatrix}$$

$$= \begin{bmatrix} 0.0224330125 & -0.0003945405 & 0.0000654469 \\ -0.0003945405 & 0.0099722201 & -0.0002361998 \\ 0.0000654469 & -0.0002361998 & 0.0072418858 \end{bmatrix}$$

For this example, the transformation we're applying to our 3 survival estimates (which we'll call Y) is the product of the estimates (i.e., $\hat{Y} = \hat{\varphi}_1\hat{\varphi}_2\hat{\varphi}_3$).

Thus, our variance estimate is given as

$$\widehat{\mathrm{var}}(\hat{Y}) \approx \left[\left(\frac{\partial(\hat{Y})}{\partial\hat{\varphi}_1}\right) \quad \left(\frac{\partial(\hat{Y})}{\partial\hat{\varphi}_2}\right) \quad \left(\frac{\partial(\hat{Y})}{\partial\hat{\varphi}_3}\right)\right] \cdot \widehat{\sum} \cdot \begin{bmatrix} \left(\frac{\partial(\hat{Y})}{\partial\hat{\varphi}_1}\right) \\ \left(\frac{\partial(\hat{Y})}{\partial\hat{\varphi}_2}\right) \\ \left(\frac{\partial(\hat{Y})}{\partial\hat{\varphi}_3}\right) \end{bmatrix}$$

Each of the partial derivatives for $\hat{Y}$ is easy enough to derive for this example. Since $\hat{Y} = \hat{\varphi}_1\hat{\varphi}_2\hat{\varphi}_3$, then $\partial\hat{Y}/\partial\hat{\varphi}_1 = \hat{\varphi}_2\hat{\varphi}_3$. And so on.

So,

$$\widehat{\mathrm{var}}(\hat{Y}) \approx \left[\left(\frac{\partial(\hat{Y})}{\partial\hat{\varphi}_1}\right) \quad \left(\frac{\partial(\hat{Y})}{\partial\hat{\varphi}_2}\right) \quad \left(\frac{\partial(\hat{Y})}{\partial\hat{\varphi}_3}\right)\right] \cdot \widehat{\sum} \cdot \begin{bmatrix} \left(\frac{\partial(\hat{Y})}{\partial\hat{\varphi}_1}\right) \\ \left(\frac{\partial(\hat{Y})}{\partial\hat{\varphi}_2}\right) \\ \left(\frac{\partial(\hat{Y})}{\partial\hat{\varphi}_3}\right) \end{bmatrix}$$

* The variance-covariance estimates **MARK** generates will occasionally depend somewhat on which optimization method you use (i.e., default, or simulated annealing), and (occasionally) on the starting values used to initialize the optimization. The differences in the reported values are often very small (i.e., apparent only several decimal places out from zero), but you should be aware of them. For all of the examples presented in this Appendix, we have used the default optimization routines, and default starting values.

$$= \begin{bmatrix} (\hat{\varphi}_2\hat{\varphi}_3) & (\hat{\varphi}_1\hat{\varphi}_3) & (\hat{\varphi}_1\hat{\varphi}_2) \end{bmatrix} \cdot \begin{bmatrix} \widehat{\text{var}}(\hat{\varphi}_1) & \widehat{\text{cov}}(\hat{\varphi}_1, \hat{\varphi}_2) & \widehat{\text{cov}}(\hat{\varphi}_1, \hat{\varphi}_3) \\ \widehat{\text{cov}}(\hat{\varphi}_1, \hat{\varphi}_1) & \widehat{\text{var}}(\hat{\varphi}_2) & \widehat{\text{cov}}(\hat{\varphi}_2, \hat{\varphi}_3) \\ \widehat{\text{cov}}(\hat{\varphi}_3, \hat{\varphi}_1) & \widehat{\text{cov}}(\hat{\varphi}_3, \hat{\varphi}_2) & \widehat{\text{var}}(\hat{\varphi}_3) \end{bmatrix} \cdot \begin{bmatrix} (\hat{\varphi}_2\hat{\varphi}_3) \\ (\hat{\varphi}_1\hat{\varphi}_3) \\ (\hat{\varphi}_1\hat{\varphi}_2) \end{bmatrix}$$

Clearly, this expression is getting more and more 'impressive' as we progress. Here is the resulting expression (written in piecewise fashion to make it easier to see the basic pattern):

$$\begin{aligned} \widehat{\text{var}}(\hat{Y}) \approx\ & \hat{\varphi}_2^2\hat{\varphi}_3^2[\widehat{\text{var}}(\hat{\varphi}_1)] \\ & + 2\hat{\varphi}_2\hat{\varphi}_3^2\hat{\varphi}_1[\widehat{\text{cov}}(\hat{\varphi}_1, \hat{\varphi}_2)] \\ & + 2\hat{\varphi}_2^2\hat{\varphi}_3\hat{\varphi}_1[\widehat{\text{cov}}(\hat{\varphi}_1, \hat{\varphi}_3)] \\ & + \hat{\varphi}_1^2\hat{\varphi}_3^2[\widehat{\text{var}}(\hat{\varphi}_2)] \\ & + 2\hat{\varphi}_1^2\hat{\varphi}_3\hat{\varphi}_2[\widehat{\text{cov}}(\hat{\varphi}_2, \hat{\varphi}_3)] \\ & + \hat{\varphi}_1^2\hat{\varphi}_2^2[\widehat{\text{var}}(\hat{\varphi}_3)] \end{aligned}$$

Whew - a lot of work (and if you think this equation looks 'impressive', try it using a second-order Taylor series approximation!). But, under some assumptions, the Delta method does rather well in allowing you to derive an estimate of the variance for functions of random variables (or, as we've described, functions of estimated parameters).

After substituting in our estimates for φ_i and the variances and covariances, our estimate for the variance of the product $\hat{Y} = (\hat{\varphi}_1\hat{\varphi}_2\hat{\varphi}_3)$ is (approximately) $\widehat{\text{var}}(Y) = 0.0025565$.

Example (2) - variance of estimate of reporting rate

In some cases animals are tagged or banded to estimate a "*reporting rate*" – the proportion of tagged animals reported (say, to a conservation management agency), given that they were killed and retrieved by a hunter or angler (see chapter 8 for more details). Thus, N_c animals are tagged with normal (*control*) tags and, of these, R_c are recovered the first year following release. The *recovery rate* of these control animals is merely R_c/N_c and we denote this as f_c.

Another group of animals, of sample size N_r, are tagged with special *reward* tags; these tags indicate that some amount of money (say, \$50) will be given to people reporting these special tags. It is assumed that \$50 is sufficient to ensure that all such tags will be reported, thus these serve as a basis for comparison and the estimation of a reporting rate. The recovery probability for the reward tagged animals is merely R_r/N_r, where R_r is the number of recoveries of reward-tagged animals the first year following release. We denote this recovery probability as f_r.

The estimator of the *reporting rate* is a ratio of the *recovery rates* and we denote this as λ. Thus:

$$\hat{\lambda} = \frac{\hat{f}_c}{\hat{f}_r}$$

Now, note that both recovery probabilities are binomials.

Thus:

$$\widehat{\text{var}}(\hat{f}_c) = \frac{\hat{f}_c\left(1-\hat{f}_c\right)}{N_c}$$

$$\widehat{\text{var}}(\hat{f}_r) = \frac{\hat{f}_r\left(1-\hat{f}_r\right)}{N_r}$$

In this case, the samples are independent, thus $\text{cov}(f_c, f_r)$ and the sampling variance-covariance matrix is diagonal:

$$\begin{bmatrix} \widehat{\text{var}}(\hat{f}_c) & 0 \\ 0 & \widehat{\text{var}}(\hat{f}_r) \end{bmatrix}$$

Next, we need the derivatives of λ with respect to f_c and f_r:

$$\frac{\partial\hat{\lambda}}{\partial\hat{f}_c} = \frac{1}{\hat{f}_r} \qquad \frac{\partial\hat{\lambda}}{\partial\hat{f}_r} = -\frac{\hat{f}_c}{\hat{f}_r^2}$$

Thus,

$$\widehat{\text{var}}(\hat{\lambda}) \approx \begin{bmatrix} \frac{1}{\hat{f}_r}, -\frac{\hat{f}_c}{\hat{f}_r^2} \end{bmatrix} \begin{bmatrix} \widehat{\text{var}}(\hat{f}_c) & 0 \\ 0 & \widehat{\text{var}}(\hat{f}_r) \end{bmatrix} \begin{bmatrix} \frac{1}{\hat{f}_r} \\ -\frac{\hat{f}_c}{\hat{f}_r^2} \end{bmatrix}$$

Example (3) - variance of back-transformed estimates - simple

In Chapter 6, we demonstrated how we can 'back-transform' from the estimate of β on the logit scale to an estimate of some parameter θ (e.g., φ or p) on the probability scale (which is bounded $[0, 1]$). But, we're clearly also interested in an estimate of the variance (precision) of our estimate, on both scales. Your first thought might be to simply back-transform from the link function (in our example, the logit link), to the probability scale, just as we did above. But, as discussed in chapter 6, this does not work.

For example, consider the male Dipper data. Using the logit link, we fit the time-invariant model $\{\varphi_{.}\, p_{.}\}$ to the data. Let's consider only the estimate for $\hat{\varphi}$. The estimate for $\hat{\beta}$ for φ is 0.2648275. Thus, our estimate of $\hat{\varphi}$ on the probability scale (which is what **MARK** reports) is:

$$\hat{\varphi} = \frac{e^{0.2648275}}{1+e^{0.2648275}} = \frac{1.303206}{2.303206} = 0.5658226$$

But, what about the variance? Well, if we look at the β estimates, **MARK** reports that the standard error for the estimate of β corresponding to survival is 0.1446688. If we simply back-transform this from the logit scale to the probability scale, we get:

$$\widehat{\text{SE}} = \frac{e^{0.1446688}}{1+e^{0.1446688}} = \frac{1.155657}{2.155657} = 0.5361043$$

However, **MARK** reports the estimated standard error for φ as 0.0355404, which isn't even remotely close to our back-transformed value of 0.5361043.

What has happened? Well, hopefully you now realize that you're 'transforming' the estimate from one scale (logit) to another (probability). And, since you're working with a 'transformation', you need to use the Delta method to estimate the variance of the back-transformed parameter.

Since

$$\hat{\varphi} = \frac{e^{\hat{\beta}}}{1 + e^{\hat{\beta}}},$$

then

$$\begin{aligned}
\widehat{\mathrm{var}}(\hat{\varphi}) &\approx \left(\frac{\partial \hat{\varphi}}{\partial \hat{\beta}}\right)^2 \times \widehat{\mathrm{var}}(\hat{\beta}) \\
&= \left(\frac{e^{\hat{\beta}}}{1 + e^{\hat{\beta}}} - \frac{\left(e^{\hat{\beta}}\right)^2}{1 + \left(e^{\hat{\beta}}\right)^2}\right)^2 \times \widehat{\mathrm{var}}(\hat{\beta}) \\
&= \left(\frac{e^{\hat{\beta}}}{\left(1 + e^{\hat{\beta}}\right)^2}\right)^2 \times \widehat{\mathrm{var}}(\hat{\beta})
\end{aligned}$$

It is worth noting that if

$$\hat{\varphi} = \frac{e^{\hat{\beta}}}{1 + e^{\hat{\beta}}},$$

then it can be easily shown that

$$\hat{\varphi}(1 - \hat{\varphi}) = \frac{e^{\hat{\beta}}}{(1 + e^{\hat{\beta}})^2},$$

which is the derivative of φ with respect to β.

So, we could rewrite our expression for the variance of $\hat{\varphi}$ conveniently as

$$\widehat{\mathrm{var}}(\hat{\varphi}) \approx \left(\frac{e^{\hat{\beta}}}{\left(1 + e^{\hat{\beta}}\right)^2}\right)^2 \times \widehat{\mathrm{var}}(\hat{\beta}) = \left(\hat{\varphi}(1 - \hat{\varphi})\right)^2 \times \widehat{\mathrm{var}}(\hat{\beta})$$

From **MARK**, the estimate of the SE for $\hat{\beta}$ was 0.1446688. Thus, the estimate of var(β) is $(0.1446688)^2$ = 0.02092906. Given the estimate of $\hat{\beta}$ of 0.2648275, we substitute into the preceding expression, which yields

$$\begin{aligned}
\widehat{\mathrm{var}}(\hat{\varphi}) &\approx \left(\frac{e^{\hat{\beta}}}{\left(1 + e^{\hat{\beta}}\right)^2}\right)^2 \times \widehat{\mathrm{var}}(\hat{\beta}) \\
&= 0.0603525 \times 0.02092906 = 0.001263
\end{aligned}$$

So, the estimated SE for $\hat{\varphi}$ is $\sqrt{0.001263}$ = 0.0355404, which is what is reported by **MARK**.

begin sidebar

SE and 95% CI

The standard approach to calculating 95% confidence limits for some parameter θ is $\theta \pm (1.96 \times \mathrm{SE})$. Is this how **MARK** calculates the 95% CI on the real probability scale? Well, take the example we just considered - the estimated SE for $\hat{\varphi} = 0.5658226$ was $\sqrt{0.001263} = 0.0355404$. So, you might assume that the 95% CI on the real probability scale would be $0.5658226 \pm (2 \times 0.0355404)$ - [0.4947418, 0.6369034].

However, this is not what is reported by **MARK** - [0.4953193, 0.6337593], which is quite close, but not exactly the same. Why the difference? The difference is because **MARK** first calculated the 95% CI on the logit scale, before back-transforming to the real probability scale. So, for our estimate of $\hat{\varphi}$, the 95% CI on the logit scale for $\hat{\beta} = 0.2648275$ is [−0.0187234, 0.5483785], which, when back-transformed to the real probability scale is [0.4953193, 0.6337593], which is what is reported by **MARK**. In this case, the very small difference between the two CI's is because the parameter estimate was quite close to 0.5. In such cases, not only will the 95% CI be nearly the same (for estimates of 0.5, it will be identical), but they will also be symmetrical.

However, because the logit transform is not linear, the *reconstituted* 95% CI will not be symmetrical around the parameter estimate, especially for parameters estimated near the [0, 1] boundaries. For example, consider the estimate for $\hat{p} = 0.9231757$. On the logit scale, the 95% CI for the β corresponding to p ($\widehat{\mathrm{SE}} = 0.5120845$) is [1.4826128, 3.4899840]. The back-transformed CI is [0.8149669, 0.9704014], which is what is reported by **MARK**. This CI is clearly **not** symmetric around $\hat{p} = 0.9231757$. The degree of asymmetry is a function of how close the estimated parameter is to either the 0 or 1 boundary. Further, the estimated variance for $\hat{p}$:

$$\begin{aligned}\widehat{\mathrm{var}}(\hat{p}) &\approx [\hat{p}(1-\hat{p})]^2 \times \widehat{\mathrm{var}}(\hat{\beta}) \\ &= [0.9231757(1-0.9231757)]^2 \times 0.262231 \\ &= 0.001319,\end{aligned}$$

yields an estimated SE of 0.036318 on the normal probability scale (which is what is reported by **MARK**). Estimating the 95% CI on the probability scale simply as $0.9231757 \pm (2 \times 0.036318)$ yields [0.85054, 0.99581], which is clearly quite a bit different, and more symmetrical, than what is reported by **MARK** (from above, [0.8149669, 0.9704014]). **MARK** uses the back-transformed CI to ensure that the reported CI is bounded [0, 1]. As the estimated parameter approaches either the 0 or 1 boundary, the degree of asymmetry in the back-transformed 95% CI that **MARK** reports will increase.

end sidebar

Example (4) - variance of back-transformed estimates - harder

In Chapter 6 we considered the analysis of variation in the survival of the European Dipper, as a function of whether or not there was a flood in the sampling area. Here, we consider just the male Dipper data (the encounter data are contained in `ed_males.inp`). Recall that for these data, there are 7 sampling occasions (6 intervals), and that a flood occurred during over the second and third intervals. For convenience, we'll assume that encounter probability is constant over time, and that survival is a linear function of '`flood`' or '`non-flood`'.

Using a logit link function, where '`flood`' years were coded in the design matrix using a '`1`', and '`non-flood`' years were coded using a '`0`', the estimated linear model for survival on the logit scale was:

$$\mathrm{logit}(\hat{\varphi}) = 0.4267863 - 0.5066372(\mathtt{flood})$$

So, in a flood year:

$$\begin{aligned}\text{logit}(\hat{\varphi}_{\text{flood}}) &= 0.4267863 - 0.5066372(\texttt{flood}) \\ &= 0.4267863 - 0.5066372(1) = -0.0798509\end{aligned}$$

Back-transforming onto the real probability scale:

$$\hat{\varphi}_{\text{flood}} = \frac{e^{-0.0798509}}{1 + e^{-0.0798509}} = 0.48005,$$

which is precisely what is reported by **MARK**.

Now, what about the estimated variance for φ_{flood}? First, what is our 'transformation function' (Y)? Simple – it is the 'back-transform' of the linear equation on the logit scale.

Given that:

$$\begin{aligned}\text{logit}(\hat{\varphi}) &= \beta_1 + \beta_2(\texttt{flood}) \\ &= 0.4267863 - 0.5066372(\texttt{flood}),\end{aligned}$$

then the back-transform function Y is

$$\hat{Y} = \frac{e^{0.4267863 - 0.5066372(\texttt{flood})}}{1 + e^{0.4267863 - 0.5066372(\texttt{flood})}}$$

Second, since our transformation clearly involves multiple parameters (β_1, β_2), the estimate of the variance is given to first-order by

$$\begin{aligned}\widehat{\text{var}}(\hat{Y}) &\approx \mathbf{D}\boldsymbol{\Sigma}\mathbf{D}^{\mathrm{T}} \\ &= \left[\frac{\partial(\hat{Y})}{\partial(\hat{\theta})}\right] \cdot \widehat{\sum} \cdot \left[\frac{\partial(\hat{Y})}{\partial(\hat{\theta})}\right]^{\mathrm{T}}\end{aligned}$$

Given our linear (transformation) equation, then the vector of partial derivatives is (we've transposed it to make it easily fit on the page):

$$\left[\left(\frac{\partial(\hat{Y})}{\partial\hat{\beta}_1}\right) \quad \left(\frac{\partial(\hat{Y})}{\partial\hat{\beta}_2}\right)\right]^{\mathrm{T}}$$

$$= \begin{bmatrix} \dfrac{e^{\beta_1 + \beta_2(\texttt{flood})}}{1 + e^{\beta_1 + \beta_2(\texttt{flood})}} - \dfrac{\left(e^{\beta_1 + \beta_2(\texttt{flood})}\right)^2}{\left(1 + e^{\beta_1 + \beta_2(\texttt{flood})}\right)^2} \\ \dfrac{\texttt{flood} \times e^{\beta_1 + \beta_2(\texttt{flood})}}{1 + e^{\beta_1 + \beta_2(\texttt{flood})}} - \dfrac{\texttt{flood} \times \left(e^{\beta_1 + \beta_2(\texttt{flood})}\right)^2}{\left(1 + e^{\beta_1 + \beta_2(\texttt{flood})}\right)^2} \end{bmatrix}$$

While this is fairly 'ugly' looking, the structure is quite straightforward - the only difference between the 2 elements of the vector is that the numerator of both terms (on either side of the minus sign) are

multiplied by 1, and `flood`, respectively. Where do these scalar multipliers come from? They're simply the partial derivatives of the linear model (we'll call it Y) on the logit scale:

$$Y = \text{logit}(\hat{\varphi}) = \beta_1 + \beta_2(\texttt{flood}),$$

with respect to each of the parameters (β_i) in turn. In other words, $\partial Y/\partial\beta_1 = 1$, and $\partial Y/\partial\beta_2 = \texttt{flood}$.

Substituting in our estimates for $\hat{\beta}_1 = 0.4267863$ and $\hat{\beta}_2 = -0.5066372$, and setting `flood=1` (to indicate a 'flood year') yields:

$$\left[\left(\frac{\partial(\hat{Y})}{\partial\hat{\beta}_1}\right) \quad \left(\frac{\partial(\hat{Y})}{\partial\hat{\beta}_2}\right)\right] = \begin{bmatrix}0.249602 & 0.249602\end{bmatrix}$$

From the **MARK** output (after exporting to a dBase file - and **not** to the Notepad - in order to get full precision), the full V-C matrix for the parameters β_1 and β_2 is:

$$\widehat{\text{cov}}(\hat{\beta}_1, \hat{\beta}_2) = \widehat{\sum} = \begin{bmatrix} 0.0321405326 & -0.0321581167 \\ -0.0321581167 & 0.0975720877 \end{bmatrix}$$

So,

$$\widehat{\text{var}}(\hat{Y}) \approx \begin{bmatrix}0.249602 & 0.249602\end{bmatrix} \times \begin{bmatrix} 0.0321405326 & -0.0321581167 \\ -0.0321581167 & 0.0975720877 \end{bmatrix} \times \begin{bmatrix}0.249602 \\ 0.249602\end{bmatrix}$$

$$= 0.0040742678$$

So, the estimated SE for the variance for the reconstituted value of survival for an individual during a 'flood year' is $\sqrt{0.0040742678} = 0.0638300$, which is what is reported by **MARK** (to within rounding error).

begin sidebar

Once again...SE and 95% CI

As noted in the precceding example, the standard approach to calculating 95% confidence limits for some parameter θ is $\theta \pm (1.96 \times \text{SE})$. However, to guarantee that the calculated 95% CI is $[0, 1]$ bounded for parameters that are $[0, 1]$ bounded (like φ), **MARK** first calculates the 95% CI on the logit scale, before back-transforming to the real probability scale. However, because the logit transform is not linear, the *reconstituted* 95% CI will not be symmetrical around the parameter estimate, especially for parameters estimated near the $[0, 1]$ boundaries.

For the present example, the estimated value of survival for an individual during a 'flood year' ($\hat{\varphi}_{\text{flood}} = 0.48005$), **MARK** reports a 95% CI of $[0.3586850, 0.6038121]$. But, where do the values $[0.3586850, 0.6038121]$ come from? Clearly, they are not based on $0.48005 \pm 1.96(\text{SE})$. Given $\widehat{\text{SE}} = 0.06383$, this would yield a 95% CI of $[0.35494, 0.60516]$, which is close, but not exactly what **MARK** reports.

In order to derive the 95% CI, we first need to calculate the variance (and SE) of the estimate on the logit scale. In the preceding example, this was very straightforward, since the model we considered had a single β term for the parameter of interest. Meaning, we could simply use the estimated SE for β to derive the 95% CI on the logit scale, which we then back-transformed onto the real probability scale. For the present example, however, the parameter is estimated from a function (transformation) involving more than one β term. In this example, the linear equation, which for consistency with the

preceding we will denote as Y, was written as:

$$\hat{Y} = \text{logit}(\hat{\varphi}) = \beta_1 + \beta_2(\texttt{flood})$$

Thus, the estimated variance of $\text{logit}(\hat{\varphi}_{\text{flood}})$ is approximated as

$$\begin{aligned}\widehat{\text{var}}(\hat{Y}) &\approx \mathbf{D\Sigma D}^{\text{T}} \\ &= \left[\frac{\partial(\hat{Y})}{\partial(\hat{\beta}_1)} \quad \frac{\partial(\hat{Y})}{\partial(\hat{\beta}_2)}\right] \cdot \widehat{\sum} \cdot \left[\frac{\partial(\hat{Y})}{\partial(\hat{\beta}_1)} \quad \frac{\partial(\hat{Y})}{\partial(\hat{\beta}_2)}\right]^{\text{T}}\end{aligned}$$

Since

$$\left[\frac{\partial(\hat{Y})}{\partial(\hat{\beta}_1)} \quad \frac{\partial(\hat{Y})}{\partial(\hat{\beta}_2)}\right] = [1 \quad \texttt{flood}] = [1 \quad 1]$$

and the VC matrix for $\hat{\beta}_1$ and $\hat{\beta}_2$ is

$$\widehat{\text{cov}}(\hat{\beta}_1, \hat{\beta}_2) = \widehat{\sum} = \begin{bmatrix} 0.0321405356 & -0.0321581194 \\ -0.0321581194 & 0.0975720908 \end{bmatrix}$$

then

$$\begin{aligned}\widehat{\text{var}}(\hat{Y}) &\approx \mathbf{D\Sigma D}^{\text{T}} \\ &= \begin{bmatrix}1 & 1\end{bmatrix} \begin{bmatrix} 0.0321405356 & -0.0321581194 \\ -0.0321581194 & 0.0975720908 \end{bmatrix} \begin{bmatrix}1 \\ 1\end{bmatrix} \\ &= 0.065396\end{aligned}$$

So, the $\widehat{\text{SE}}$ – on the logit scale! – is $\sqrt{0.065396} = 0.255727$. Thus, the 95% CI on the estimate on the logit scale, $\text{logit}(\hat{\varphi}_{\text{flood}}) = -0.0798509 \pm 1.96(0.255727) = [-0.581076, 0.421374]$. All that is left is to back-transform the limits on the CI to the real probability scale:

$$[-0.94321, 0.78351] \longrightarrow \left[\frac{e^{-0.581076}}{1 + e^{-0.581076}}, \quad \frac{e^{0.421374}}{1 + e^{0.421374}}\right] = [0.358685, 0.603812]$$

which is what is reported by **MARK** (to within rounding error).

end sidebar

Example (5) - variance of back-transformed estimates - harder still

In Chapter 11, we considered analysis of the effect of various functions of mass, `m`, and mass-squared, `m2`, on the survival of a hypothetical species of bird (the simulated data are in file `indcov1.inp`). The linear function relating survival to `m` and `m2`, *on the logit scale*, is:

$$\text{logit}(\hat{\varphi}) = 0.2567341 + 1.1750463(\texttt{m}_\text{s}) - 1.0554957(\texttt{m}_\text{s}^2)$$

Note that for the two mass terms in the equation, there is a small subscript 's' - reflecting the fact that these are '*standardized*' masses. Recall that we standardized the covariates by subtracting the mean of the covariate, and dividing by the standard deviation (the use of standardized or non-standardized covariates is discussed at length in Chapter 11).

Thus, for each individual in the sample, the estimated survival probability (on the logit scale) for that individual, given its mass, is given by:

$$\text{logit}(\hat{\varphi}) = 0.2567333 + 1.1750526\left(\frac{\mathtt{m} - \bar{\mathtt{m}}}{\text{SD}_{\mathtt{m}}}\right) - 1.0555024\left(\frac{\mathtt{m}^2 - \bar{\mathtt{m}}^2}{\text{SD}_{\mathtt{m}^2}}\right)$$

In this expression, `m` refers to `mass` and $\mathtt{m}^2$ refers to `mass2`. The output from **MARK** (preceding page) actually gives you the mean and standard deviations for both covariates: for `mass`, mean = 109.9680, and SD = 24.7926, while for `mass2`, the mean = 12,707.4640, and the SD = 5,532.0322. The '**`value`**' column shows the standardized values for `mass` and `mass2` (0.803 and 0.752) for the first individual in the data file.

Now let's consider a worked example of the calculation of the variance of estimated survival. Suppose the mass of the bird was 110 units, so that `m` = 110, `m2` = $(110)^2$ = 12,100.

Thus:

$$\begin{aligned}\text{logit}(\hat{\varphi}) &= 0.2567333 + 1.1750526\left(\frac{(110 - 109.9680)}{24.7926}\right) - 1.0555024\left(\frac{(12{,}100 - 12{,}707.4640)}{5{,}532.0322}\right) \\ &= 0.3742\end{aligned}$$

So, if $\text{logit}(\hat{\varphi}) = 0.374$, then the reconstituted estimate of φ, transformed back from the logit scale is:

$$\frac{e^{0.374152}}{1 + e^{0.374152}} = 0.5925$$

Thus, for an individual weighing 110 units, the expected annual survival probability is approximately 0.5925 (which is what **MARK** reports if you use the '**`User specify covariate`**' option).

What about the variance (and corresponding SE) for this estimate? First, what is our 'transformation function' (Y)? For the present example, it is the 'back-transform' of the linear equation on the logit scale. Given that:

$$\begin{aligned}\text{logit}(\hat{\varphi}) &= \beta_1 + \beta_2(\mathtt{m}_s) + \beta_3(\mathtt{m}_s^2) \\ &= 0.2567333 + 1.1750526(\mathtt{m}_s) - 1.0555024(\mathtt{m}_s^2),\end{aligned}$$

then the back-transform Y is:

$$\hat{Y} = \frac{e^{0.2567333 + 1.1750526(\mathtt{m}_s) - 1.0555024(\mathtt{m}_s^2)}}{1 + e^{0.2567333 + 1.1750526(\mathtt{m}_s) - 1.0555024(\mathtt{m}_s^2)}}$$

As in the preceding example, since our transformation clearly involves multiple parameters ($\beta_1, \beta_2, \beta_3$), the estimate of the variance is given by:

$$\begin{aligned}\widehat{\text{var}}(\hat{Y}) &\approx \mathbf{D}\boldsymbol{\Sigma}\mathbf{D}^{\mathrm{T}} \\ &= \left[\frac{\partial(\hat{Y})}{\partial(\hat{\theta})}\right] \cdot \widehat{\sum} \cdot \left[\frac{\partial(\hat{Y})}{\partial(\hat{\theta})}\right]^{\mathrm{T}}\end{aligned}$$

Given our linear (transformation) equation (from above) then the vector of partial derivatives is is:

$$\begin{bmatrix} \left(\frac{\partial(\hat{Y})}{\partial\hat{\beta}_0}\right) \\ \left(\frac{\partial(\hat{Y})}{\partial\hat{\beta}_1}\right) \\ \left(\frac{\partial(\hat{Y})}{\partial\hat{\beta}_2}\right) \end{bmatrix} = \begin{bmatrix} \frac{e^{\beta_1 + \beta_2(\mathtt{m}) + \beta_3(\mathtt{m2})}}{1 + e^{\beta_1 + \beta_2(\mathtt{m}) + \beta_3(\mathtt{m2})}} - \frac{\left[e^{\beta_1 + \beta_2(\mathtt{m}) + \beta_3(\mathtt{m2})}\right]^2}{\left[1 + e^{\beta_1 + \beta_2(\mathtt{m}) + \beta_3(\mathtt{m2})}\right]^2} \\ \frac{\mathtt{m} \times e^{\beta_1 + \beta_2(\mathtt{m}) + \beta_3(\mathtt{m2})}}{1 + e^{\beta_1 + \beta_2(\mathtt{m}) + \beta_3(\mathtt{m2})}} - \frac{\mathtt{m} \times \left[e^{\beta_1 + \beta_2(\mathtt{m}) + \beta_3(\mathtt{m2})}\right]^2}{\left[1 + e^{\beta_1 + \beta_2(\mathtt{m}) + \beta_3(\mathtt{m2})}\right]^2} \\ \frac{\mathtt{m2} \times e^{\beta_1 + \beta_2(\mathtt{m}) + \beta_3(\mathtt{m2})}}{1 + e^{\beta_1 + \beta_2(\mathtt{m}) + \beta_3(\mathtt{m2})}} - \frac{\mathtt{m2} \times \left[e^{\beta_1 + \beta_2(\mathtt{m}) + \beta_3(\mathtt{m2})}\right]^2}{\left[1 + e^{\beta_1 + \beta_2(\mathtt{m}) + \beta_3(\mathtt{m2})}\right]^2} \end{bmatrix}$$

Although this looks complicated, the structure is actually quite straightforward – the only difference between the 3 elements of the vector is that the numerator of both terms (on either side of the minus sign) are multiplied by 1, `m`, and `m2`, respectively, which are simply the partial derivatives of the linear model (we'll call it Y) on the logit scale:

$$\hat{Y} = \text{logit}(\hat{\varphi}) = \beta_1 + \beta_2(\mathtt{m}_s) + \beta_3(\mathtt{m}_s^2),$$

with respect to each of the parameters (β_i) in turn. In other words, $\partial Y/\partial\beta_1 = 1$, $\partial Y/\partial\beta_2 = \mathtt{m}$, and $\partial Y/\partial\beta_3 = \mathtt{m2}$.

So, now that we have our vectors of partial derivatives of the transformation function with respect to each of the parameters, we can simplify things considerably by substituting in the standardized values for `m` and `m2`, and the estimated parameter values ($\hat{\beta}_1$, $\hat{\beta}_2$, and $\hat{\beta}_3$). For a `mass` of 110 g, the standardized values for `m` and `m2` are:

$$\mathtt{m}_s = \left(\frac{110 - 109.9680}{24.7926}\right) = 0.0012895$$

$$\mathtt{m2}_s = \left(\frac{12100 - 12707.4640}{5532.0322}\right) = -0.1098085$$

The estimates for $\hat{\beta}_i$ we read directly from **MARK**: $\hat{\beta}_1 = 0.2567333$, $\hat{\beta}_2 = 1.1750526$, and $\hat{\beta}_3 = -1.0555024$.

Substituting in these estimates for $\hat{\beta}_i$ and the standardized `m` and `m2` values into our vector of partial derivatives (which we've transposed in the following to save space) yields:

$$\left[\left(\frac{\partial(\hat{Y})}{\partial\hat{\beta}_1}\right) \quad \left(\frac{\partial(\hat{Y})}{\partial\hat{\beta}_2}\right) \quad \left(\frac{\partial(\hat{Y})}{\partial\hat{\beta}_3}\right)\right]^{\mathrm{T}} = \begin{bmatrix} 0.241451 \\ 0.000311 \\ -0.026513 \end{bmatrix}$$

From the **MARK** output (after exporting to a dBase file - and **not** to the editor - in order to get full precision), the full V-C matrix for the β parameters is

$$\begin{bmatrix} 0.0009006967 & -0.0004110129 & 0.0003662771 \\ -0.0004110129 & 0.0373928740 & -0.0364291221 \\ 0.0003662771 & -0.0364291221 & 0.0362817338 \end{bmatrix}$$

So,

$$\widehat{\text{var}}(\hat{Y}) \approx \begin{bmatrix} 0.241451 & 0.000311 & -0.026513 \end{bmatrix} \times \begin{bmatrix} 0.0009006967 & -0.0004110129 & 0.0003662771 \\ -0.0004110129 & 0.0373928740 & -0.0364291221 \\ 0.0003662771 & -0.0364291221 & 0.0362817338 \end{bmatrix} \times \begin{bmatrix} 0.241451 \\ 0.000311 \\ -0.026513 \end{bmatrix}$$

$$= 0.000073867$$

So, the estimated SE for var for the reconstituted value of survival for an individual weighing 110 g is $\sqrt{0.000073867} = 0.0085946$, which is exactly what is reported by **MARK**.

It is important to remember that the estimated variance will vary depending on the mass you use - the estimate of the variance for a 110 g individual (0.000073867) will differ from the estimated variance for a (say) 120 g individual. For a 120 g individual, the standardized values of `m` and `m2` are 0.404636 and 0.3059512, respectively. Based on these values, then:

$$\left[\left(\frac{\partial(\hat{Y})}{\partial\hat{\beta}_1}\right)\left(\frac{\partial(\hat{Y})}{\partial\hat{\beta}_2}\right)\left(\frac{\partial(\hat{Y})}{\partial\hat{\beta}_3}\right)\right]^{\mathrm{T}} = \begin{bmatrix} 0.239817 \\ 0.097039 \\ 0.073372 \end{bmatrix}$$

Given the variance covariance-matrix for this model (shown above), then

$$\widehat{\text{var}}(\hat{Y}) \approx \mathbf{D\Sigma D}^{\mathrm{T}} = 0.000074246$$

Thus, the estimated SE for the reconstituted value of survival for an individual weighing 120 g is $\sqrt{0.000074246} = 0.0086166$, which again is exactly what is reported by **MARK**.

Note that this value for the SE for a 120 g individual (0.008617) differs from the SE estimated for a 110 g individual (0.008595), albeit not by much (the small difference here is because this is a very large simulated data set based on a deterministic model - see Chapter 11 for details). Since each weight would have its own estimated survival, and associated estimated variance and SE, to generate a curve showing the reconstituted values and their SE, you'd need to iteratively calculate $\mathbf{D\Sigma D}^{\mathrm{T}}$ over a range of weights. We'll leave it to you to figure out how to handle the programming if you want to do this on your own. For the less ambitious, **MARK** now has the capacity to do much of this for you - you can output the 95% CI 'data' over a range of individual covariate values to a spreadsheet (see section 11.5 in Chapter 11).

Example (6) - estimating variance + covariance in transformations

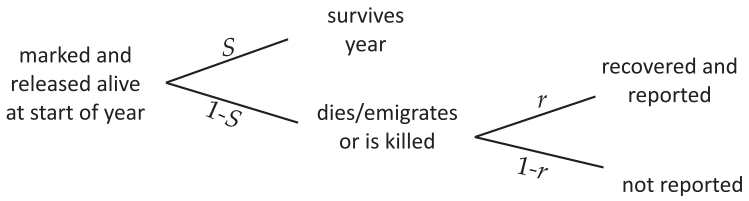
Here, we consider application of the Delta method to joint estimation of the variance of a parameter, and the *covariance* of that parameter with another, where one of the two parameters is a linear transformation of the other. This is somewhat complicated, but quite useful example, since it illustrates how you can use the Delta method to estimate not only the variance of individual parameters, but the covariance structure among parameters as well.

There are many instances where the magnitude of the covariance is of particular interest. Here, we consider such a situation, in terms of different parameterizations for analysis of dead recovery data. Dead recovery models are covered in detail in Chapter 8 – here, we briefly review two different parameterizations (the 'Seber' and 'Brownie' parameterizations), and the context of our interest in the

covariance between two different parameters.

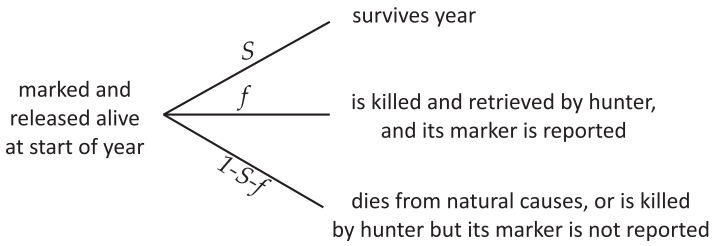
Seber (1973:254) developed a dead recovery model where marked individuals are assumed to survive from release i to $i+1$ with probability S_i. Individuals may die during the interval, either due to harvest or to 'natural' mortality. The probability that dead marked individuals are reported during each period i between releases, and (most generally) where the death is not necessarily related to harvest, is r_i. In other words, r_i is the joint probability of (i) the marked individual dying from either harvest or natural causes, and (ii) being recovered and reported (i.e., 'encountered').

The encounter process for the Seber parameterization is illustrated in the following:



Brownie *et al*. (1985) (hereafter, simply 'Brownie') developed a different parameterization for dead recovery data, where the sources of mortality (harvest, versus 'natural' or non-harvest) are modeled separately. Following Brownie, S_i = the probability that the individual survives the interval from release occasion i to $i+1$ (note that the definition for the probability of survival is logically identical between the Seber and Brownie parameterizations). The probability that the individual dies from either source of mortality is simply $1 - S$. However, in contrast to the Seber parameterization, Brownie specified a parameter f, to represent the probability that an individual dies specifically due to harvest during interval i, and is reported ('encountered'). Thus, the probability that the individuals dies from natural causes is $(1 - S - f)$.

The encounter process for the Brownie parameterization is illustrated in the following:



There is, in fact, a simple algebraic relationship between the two parameterizations (assuming the same encounter data are used). Under the Seber parameterization, the probability of the encounter history '11' is given as $r(1-S)$. In contrast, under the Brownie parameterization, the expected probability of this event is simply f. Since the encounter history is the same, we can set the different parameterizations for the expected probability of the event equal to each other, which leads to the following expressions relating the two parameterizations:

$$f_i = r_i(1 - S_i) \qquad r_i = \frac{f_i}{(1 - S_i)}$$

Clearly, the parameter r_i is a reduced parameter, and can be expressed as a function of two other parameters normally found in the Brownie parameterization. An obvious practical question is, why use one parameterization over the other, and does it matter? This issue is discussed more fully in Chapter 8, but for now, we focus on the left-hand expression:

$$f_i = r_i(1 - S_i)$$

So, given estimates of $\hat{r}_i$ and $\hat{S}_i$ from a Seber analysis, we could use this algebraic relationship (i.e., transformation) to generate estimates of $\hat{f}$. Naturally, we wish to be able to estimate $\widehat{\text{var}}(\hat{f})$.

However, in addition, we are potentially interested in estimating the covariance $\widehat{\text{cov}}(\hat{f}, \hat{S})$. Recall from above that the parameter f relates in part to the probability of being harvested. We might naturally be interested in the relationship between harvest mortality f, and overall annual survival, S. For example, if harvest and natural mortality are strictly additive, then we might expect a negative covariance between survival and harvest (i.e., as the probability of mortality due to harvest increases, annual survival will decrease). Whether or not the covariance is negative has important implications for harvest management (see full discussion in the Williams, Nichols & Conroy 2001 book).

We'll begin by considering estimation of the variance for $\hat{f}$ only, using the Delta method. Let the transformation g be $f = (1 - S)r$. Given $\hat{S}, \hat{r}, \widehat{\text{var}}(\hat{S})$ and $\widehat{\text{var}}(\hat{r})$, then the Jacobian for g is

$$\begin{bmatrix} \frac{\partial g}{\partial S} & \frac{\partial g}{\partial r} \end{bmatrix} = \begin{bmatrix} -\hat{r} & 1 - \hat{S} \end{bmatrix},$$

and thus

$$\widehat{\text{var}}(\hat{f}) \approx \begin{bmatrix} -\hat{r} & 1 - \hat{S} \end{bmatrix} \cdot \widehat{\sum} \cdot \begin{bmatrix} -\hat{r} \\ 1 - \hat{S} \end{bmatrix},$$

where $\widehat{\sum}$ is the variance-covariance matrix for S and r:

$$\widehat{\sum} = \begin{bmatrix} \widehat{\text{var}}(\hat{S}) & \widehat{\text{cov}}(\hat{S}, \hat{r}) \\ \widehat{\text{cov}}(\hat{S}, \hat{r}) & \widehat{\text{var}}(\hat{r}) \end{bmatrix}.$$

So,

$$\widehat{\text{var}}(\hat{f}) \approx \begin{bmatrix} -\hat{r} & 1 - \hat{S} \end{bmatrix} \cdot \widehat{\sum} \cdot \begin{bmatrix} -\hat{r} \\ 1 - \hat{S} \end{bmatrix}$$

yields

$$\widehat{\text{var}}(\hat{f}) \approx \hat{r}^2 \widehat{\text{var}}(\hat{S}) - 2\hat{r} \cdot \widehat{\text{cov}}(\hat{S}, \hat{r}) + 2\hat{r} \cdot \widehat{\text{cov}}(\hat{S}, \hat{r})\hat{S} + \widehat{\text{var}}(\hat{r}) - 2 \cdot \widehat{\text{var}}(\hat{r})S + \widehat{\text{var}}(\hat{r})S^2$$

which, with a little re-arranging, yields

$$\widehat{\text{var}}(\hat{f}) \approx \hat{r}^2 \cdot \widehat{\text{var}}(\hat{S}) - 2[(1 - \hat{S})\hat{r}] \cdot \widehat{\text{cov}}(\hat{S}, \hat{r}) + (1 - \hat{S})^2 \cdot \widehat{\text{var}}(\hat{r}),$$

If you substitute in $r = f/(1 - S)$ into the preceding expression, we end up with

$$\widehat{\text{var}}(\hat{f}) \approx \left(\frac{\hat{f}}{1-\hat{S}}\right)^2 \cdot \widehat{\text{var}}(\hat{S}) - 2\hat{f} \cdot \widehat{\text{cov}}(\hat{S}, \hat{r}) + (1-\hat{S})^2 \cdot \widehat{\text{var}}(\hat{r}).$$

Now, what if instead of $\widehat{\text{var}}(\hat{f})$ only, we are also interested in estimating the *covariance* of (say) f and S? Such a covariance might be of interest since f is a function of S, and there may be interest in the degree to which S varies as a function of f (see above). Thus, we want to apply the Delta method to a function (the covariance) of two parameters, f and S. The key step here is recognizing that there are in fact two different functions (or, transformations) involved, which we'll call g_1 and g_2:

$$g_1 : S \rightarrow S \qquad \text{and} \qquad g_2 : (1-S)r \rightarrow f$$

You might be puzzled by $g_1 : S \rightarrow S$. In fact, this represents a null transformation – a direct, non-transformative 1:1 mapping between S under the Seber parameterization, and survival under the Brownie parameterization (since the probability of surviving is, logically, the same under the two parameterizations). This is analogous to generating the estimate for $\hat{S}_i$ under one paramterization by multiplying the same estimate under the other parameterization by the scalar constant 1.

Thus, with two transformations, we generate a Jacobian *matrix* of partial derivatives of each transformations with respect to S and r, respectively:

$$\begin{aligned}
\begin{bmatrix} \frac{\partial g_1}{\partial \hat{S}} & \frac{\partial g_1}{\partial \hat{r}} \\ \frac{\partial g_2}{\partial \hat{S}} & \frac{\partial g_2}{\partial \hat{r}} \end{bmatrix} &= \begin{bmatrix} \frac{\partial \hat{S}}{\partial \hat{S}} & \frac{\partial \hat{S}}{\partial \hat{r}} \\ \frac{\partial \hat{f}}{\partial \hat{S}} & \frac{\partial \hat{f}}{\partial \hat{r}} \end{bmatrix} \\
&= \begin{bmatrix} 1 & 0 \\ -r & 1-\hat{S} \end{bmatrix} \\
&= \begin{bmatrix} 1 & 0 \\ -\frac{\hat{f}}{(1-\hat{S})} & 1-\hat{S} \end{bmatrix}.
\end{aligned}$$

Given the variance-covariance matrix $\widehat{\sum}$ for $\hat{S}$ and $\hat{r}$

$$\widehat{\sum} = \begin{bmatrix} \widehat{\text{var}}(\hat{S}) & \widehat{\text{cov}}(\hat{S}, \hat{r}) \\ \widehat{\text{cov}}(\hat{S}, \hat{r}) & \widehat{\text{var}}(\hat{r}) \end{bmatrix},$$

we evaluate sampling variance-covariance matrix for $\hat{S}$ and $\hat{f}$ as the matrix product

$$\begin{bmatrix} 1 & 0 \\ -\frac{\hat{f}}{(1-\hat{S})} & 1-\hat{S} \end{bmatrix} \cdot \widehat{\sum} \cdot \begin{bmatrix} 1 & -\frac{\hat{f}}{(1-\hat{S})} \\ 0 & 1-\hat{S} \end{bmatrix},$$

which (after a bit of algebra) yields

$$\begin{bmatrix} \widehat{\text{var}}(\hat{S}) & -\dfrac{\hat{f}}{1-\hat{S}} \cdot \widehat{\text{var}}(\hat{S}) + (1-\hat{S}) \cdot \widehat{\text{cov}}(\hat{S},\hat{r}) \\ -\dfrac{\hat{f}}{1-\hat{S}} \cdot \widehat{\text{var}}(\hat{S}) + (1-\hat{S}) \cdot \widehat{\text{cov}}(\hat{S},\hat{r}) & \hat{r}^2 \cdot \widehat{\text{var}}(\hat{S}) - 2[(1-\hat{S})\hat{r}] \cdot \widehat{\text{cov}}(\hat{S},\hat{r}) + (1-\hat{S})^2 \cdot \widehat{\text{var}}(\hat{r}) \end{bmatrix}.$$

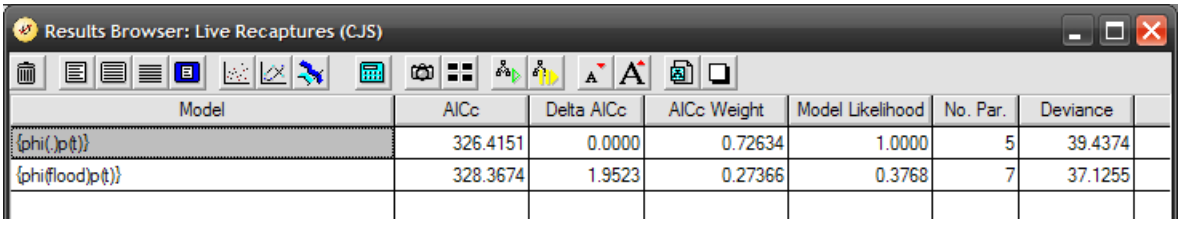
Here, matrix elements [1,1] and [2,2] are the expressions for the approximate variance of S and f, respectively (note that the expression in element [2,2], for $\widehat{\text{var}}(\hat{f})$, is identical to the expression we derived on the preceding page). Elements [1,2] and [2,1] (which are the same) are the expressions for the approximate *covariance* of f and S.

As noted earlier, interpretation of the estimated variance and covariance is dependent on the source of the variance-covariance matrix, $\widehat{\Sigma}$, used in the calculations. If $\widehat{\Sigma}$ is constructed using variances and covariances from the usual ML parameter estimates, then the resulting estimate for variance is an estimate of the *total* variance (i.e., sampling + process, where process variation represents the underlying 'biological' variation). In contrast, if $\widehat{\Sigma}$ is based on estimated *process* (variances and covariances only, then the estimate for variance is an estimate of the *process* variance. Decomposition of total variance into sampling and process components is covered in detail in Appendix D.

B.5. Delta method and model averaging

In the preceding examples, we focused on the application of the Delta method to transformations of parameter estimates from a single model. However, as introduced in Chapter 4 - and emphasized throughout the remainder of this book - we're often interested in accounting for model selection uncertainty by using model-averaged values. There is no major complication for application of the Delta method to model-averaged parameter values - you simply need to make sure you use model-averaged values for each element of the calculations.

We'll demonstrate this using analysis of the male dipper data (`ed_male.inp`). Suppose that we fit 2 candidate models to these data: $\{\varphi_{.} p_t\}$ and $\{\varphi_{flood} p_t\}$. In other words, a model where survival is constant over time, and a model where survival is constrained to be a function of a binary '`flood`' variable (see section 6.4 of Chapter 6). Here are the results of fitting these 2 models to the data:



Model	AICc	Delta AICc	AICc Weight	Model Likelihood	No. Par.	Deviance
{phi(.)p(t)}	326.4151	0.0000	0.72634	1.0000	5	39.4374
{phi(flood)p(t)}	328.3674	1.9523	0.27366	0.3768	7	37.1255

As expected (based on the analysis of these data presented in Chapter 6), we see that there is some evidence of model selection uncertainty - the model where survival is constant over time has roughly 2-3 times the weight as does a model where survival is constrained to be a function of the binary 'flood' variable.

The model averaged values for each interval are shown below:

	1	2	3	4	5	6
estimate	0.5673	0.5332	0.5332	0.5673	0.5673	0.5673
SE	0.0441	0.0581	0.0581	0.0441	0.0441	0.0441

Now, suppose we want to derive the best estimate of the probability of survival over (say) the first 3 intervals. Clearly, all we need to do is take the product of the 3 model-averaged values corresponding to the first 3 intervals:

$$(0.5673 \times 0.5332 \times 0.5332) = 0.1613$$

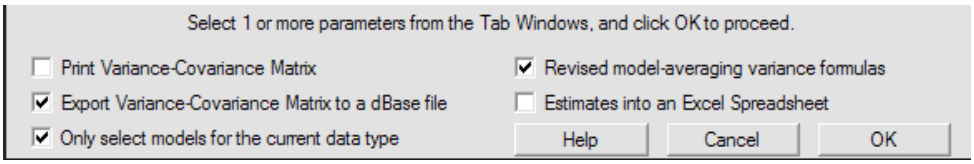
In other words, our best estimate of the probability that a male dipper would survive from the start of the time series to the end of the third interval is 0.1613.

What about the standard error of this product? Here, we use the Delta method. Recall that:

$$\begin{aligned}\widehat{\text{var}}(\hat{Y}) &\approx \mathbf{D\Sigma D}^{\mathrm{T}} \\ &= \left[\frac{\partial(\hat{Y})}{\partial(\hat{\theta})}\right] \cdot \widehat{\sum} \cdot \left[\frac{\partial(\hat{Y})}{\partial(\hat{\theta})}\right]^{\mathrm{T}},\end{aligned}$$

where Y is some linear or nonlinear function of the parameter estimates $\hat{\theta}_1, \hat{\theta}_2, \ldots$. For this example, Y is the product of the survival estimates.

So, the first thing we need to do is to generate the estimated variance-covariance matrix for the model averaged survival estimates. This is easy enough to do - in the '**`Model Averaging Parameter Selection`**' window, you simply need to '**`Export Variance-Covariance Matrix to a dBase file`**' - you do this by checking the appropriate check box (lower-left, as shown below).



The 'rounded' values which would be output to the Notepad (or whatever editor you've specified) are shown below.

```
Unconditional Variance-Covariance Matrix of Model Averaged Estimates
    Variance-Covariance matrix of estimates on diagonal and below,
            Correlation matrix of estimates above diagonal.
    |    1           2           3
----+-----------------------------------
    |
  1 |  0.00194     0.04923     0.04923
    |
  2 |  0.00013     0.00337     1.00000
    |
  3 |  0.00013     0.00337     0.00337
```

Recall that the variance-covariance matrix of estimates is given on the diagonal and below (whereas the correlation matrix of the estimates is shown above the diagonal). (*Note*: remember that for the actual calculations you need the full precision variance-covariance matrix from the exported dBase file).

All that remains is to substitute our model-averaged estimates for (i) $\hat{\varphi}$ and (ii) the variance-covariance matrix (above), into $\widehat{\text{var}}(\hat{Y}) \approx \mathbf{D\Sigma D}^{\mathrm{T}}$.

Thus,

$$\widehat{\text{var}}(\hat{Y}) \approx \left[\frac{\partial(\hat{Y})}{\partial(\hat{\theta})}\right] \cdot \widehat{\sum} \cdot \left[\frac{\partial(\hat{Y})}{\partial(\hat{\theta})}\right]^{\mathrm{T}}$$

$$= \begin{bmatrix}(\bar{\hat{\varphi}}_2\bar{\hat{\varphi}}_3) & (\bar{\hat{\varphi}}_1\bar{\hat{\varphi}}_3) & (\bar{\hat{\varphi}}_1\bar{\hat{\varphi}}_2)\end{bmatrix} \cdot \widehat{\sum} \cdot \begin{bmatrix}(\bar{\hat{\varphi}}_2\bar{\hat{\varphi}}_3)\\(\bar{\hat{\varphi}}_1\bar{\hat{\varphi}}_3)\\(\bar{\hat{\varphi}}_1\bar{\hat{\varphi}}_2)\end{bmatrix}$$

$$= \begin{bmatrix}(\bar{\hat{\varphi}}_2\bar{\hat{\varphi}}_3) & (\bar{\hat{\varphi}}_1\bar{\hat{\varphi}}_3) & (\bar{\hat{\varphi}}_1\bar{\hat{\varphi}}_2)\end{bmatrix} \cdot \begin{bmatrix}\widehat{\text{var}}(\bar{\hat{\varphi}}_1) & \widehat{\text{cov}}(\bar{\hat{\varphi}}_1,\bar{\hat{\varphi}}_2) & \widehat{\text{cov}}(\bar{\hat{\varphi}}_1,\bar{\hat{\varphi}}_3)\\ \widehat{\text{cov}}(\bar{\hat{\varphi}}_1,\bar{\hat{\varphi}}_2) & \widehat{\text{var}}(\bar{\hat{\varphi}}_2) & \widehat{\text{cov}}(\bar{\hat{\varphi}}_2,\bar{\hat{\varphi}}_3)\\ \widehat{\text{cov}}(\bar{\hat{\varphi}}_3,\bar{\hat{\varphi}}_1) & \widehat{\text{cov}}(\bar{\hat{\varphi}}_3,\bar{\hat{\varphi}}_2) & \widehat{\text{var}}(\bar{\hat{\varphi}}_3)\end{bmatrix} \cdot \begin{bmatrix}(\bar{\hat{\varphi}}_2\bar{\hat{\varphi}}_3)\\(\bar{\hat{\varphi}}_1\bar{\hat{\varphi}}_3)\\(\bar{\hat{\varphi}}_1\bar{\hat{\varphi}}_2)\end{bmatrix}$$

$$= \begin{bmatrix}0.284303069 & 0.3024783390 & 0.3024783390\end{bmatrix} \times \begin{bmatrix}0.0019410083 & 0.0001259569 & 0.0001259569\\ 0.0001259569 & 0.0033727452 & 0.0033727423\\ 0.0001259569 & 0.0033727423 & 0.0033727452\end{bmatrix} \times \begin{bmatrix}0.284303069\\0.3024783390\\0.3024783390\end{bmatrix}$$

$$= 0.001435$$

B.6. Summary

In this appendix, we've briefly introduced a convenient, generally straightforward method for deriving an estimate of the sampling variance for transformations of one or more variables. Such transformations are quite commonly encountered when using **MARK**, and having a method to derive estimates of the sampling variances is convenient. The most straightforward method – based on a first-order Taylor series expansion – is known generally as the 'Delta method'. However, the first-order Taylor series approximation may not always be appropriate, especially if the transformation is highly non-linear, and if there is significant variation in the data. In such case, you may have to resort to higher-order approximations, or numerically intensive bootstrapping approaches.

APPENDIX C

RMark - an alternative approach to building linear models in MARK

Jeff Laake
Alaska Fisheries Science Center
National Marine Fisheries Service
Seattle, Washington, USA

Eric Rexstad
Research Unit for Wildlife Population Assessment
CREEM - University of St. Andrews
St. Andrews, Scotland

For most of the examples presented in the **MARK** book, the construction of the design matrix (DM) using the 'graphical DM template' is relatively straightforward. Moreover, by 'forcing' you to confront the actual structure of the design matrix, the relationship between linear models, covariates, even fundamental statistical entities like 'degrees of freedom' may actually make more sense than they did before. However, quite often in real-world situations, where the size of design matrices can get very large, very quickly, it is often cumbersome to build design matrices in this fashion. Further, your chances of making a mistake while building the design matrix increase in rough proportion to the size of the design matrix - compounded when the models contain significant ultrastructure. Also, if either the number of occasions or group structure changes, PIMs and DMs must be changed and this means rebuilding each model in **MARK**. Thus, automated model development is almost a necessity for researchers that monitor populations over time and are continually adding sampling occasions.

This appendix describes an alternate interface that can be used in place of **MARK**'s graphical interface to describe and run models in terms of formula (e.g., **`Phi~sex+age+time`**). The interface constructs the necessary PIMS and design matrices which automates model development. The interface creates the **MARK** input file, initiates **`mark.exe`** and then extracts the results from the output files. All of the computation for parameter estimation is done with **MARK** (**`mark.exe`**). This alternative interface is a package that has been written in **R**, a freely available statistical programming environment. Thus the package was named **RMark**. This appendix provides an introduction and description of the **RMark** interface to help you get started. The appendix does not document every function and every function argument in the package because that reference material is provided in the help file documentation that accompanies the package as described below. Instead, analyses of the dipper and swift datasets and other examples are repeated here using **RMark** to demonstrate this 'formula based' approach to specifying models.

© Cooch & White (2014)

In addition to automating model development, **RMark** has the following advantages:

1. labels for real (reconstituted) and β parameters are automatically added for ease of interpretation
2. scripts can be written to run an entire analysis and the script can be documented
3. covariate-specific real parameter estimates can be computed within **R** without re-running the analysis
4. the **R** environment is available for plotting and further computation on the results.

Examples of these advantages are given throughout this appendix.

However, there are a number of disadvantages in comparison to the existing **MARK** graphical interface. First and foremost, you need to have a rudimentary knowledge of **R** before using **RMark**. There is no getting around it and while it could be viewed as a disadvantage for **RMark**, it can also be viewed as an advantage because **R** is a very powerful statistical programming environment that can be useful for many different analysis tasks and **RMark** may be the push you need to start using **R** for all of your analyses. To help you learn **R**, we provide a very brief **R** primer at the end of this appendix, but we suggest that you also take advantage of the **R** tutorial material on the web and in various books. Even if you have a reasonable grasp of **R**, it may be useful to review the tutorial to understand how lists provide useful structures for working with models in **RMark**.

At present, another disadvantage is that the **RMark** interface does not replicate every aspect of the **MARK** interface. In particular, not every model in **MARK** is supported by **RMark**. For a complete list, refer to the `MarkModels.pdf` file that is installed in the **RMark** subdirectory (from within **MARK**, see also '`Help | Data Types`'). In addition, features such as the median $\hat{c}$ goodness-of-fit testing and random effects are not available at present. A solution is to export the model runs from **RMark** into the **MARK** interface (discussed later in this appendix).

Another subtle difference with the **MARK** interface is that all models constructed in the **RMark** interface are developed via a design matrix approach rather than coding the model structure via parameter index matrices (PIMS). The title for this appendix was chosen to reflect this aspect of the **RMark** interface. However, as of version 1.7.6 RMark can create models with an identity design matrix and you can now use the sin link as long as the formula specifies a model that can be represented by a identity matrix. Obviously, you cannot use covariates with the sin link. See section C.10 for more explanation. Even though **RMark** constructs the design matrix for you, you still need to understand the concepts described in the book about design matrices and counting parameters. Having the description of **RMark** in an appendix to this book is intentional and appropriate because initially it is best to learn to use the standard interface so that you understand what **RMark** is doing.

In manuscripts, cite this appendix for **RMark** and in describing it make sure to say something like "we used the **R** (R Development Core Team 2007) package **RMark** (Laake and Rexstad 2008) to construct models for program **MARK** (White and Burnham 1999)." Use `citation()` in **R** to get the proper citation for **R**.

If you have no experience with **R** we highly recommend that you start by reading C.1 and C.24 and either a good introductory text on **R** or the online material on the **R** home page (which is found at `http://www.r-project.org/`). Once you become moderately comfortable with **R**, read sections C.2-C.12 and follow along with the examples. After reading section C.12 you should be able to import your own data and work through the more advanced sections in C.13-C.16. Specific examples of models beyond CJS are given in section C.17-C.20 and we expect to add more sections like this in future revisions. If you want to know how to export **RMark** models to use features of the **MARK** interface see section C.21. Examples of using **R** for further computation on results like creating delta method

variances are described in C.22. If you encounter errors or problems, see C.23 for a list of common errors and suggested solutions.

Some of the examples displayed here will only work with version 1.7.3 of **RMark** or later and the December, 2007 version or later of **mark.exe**.

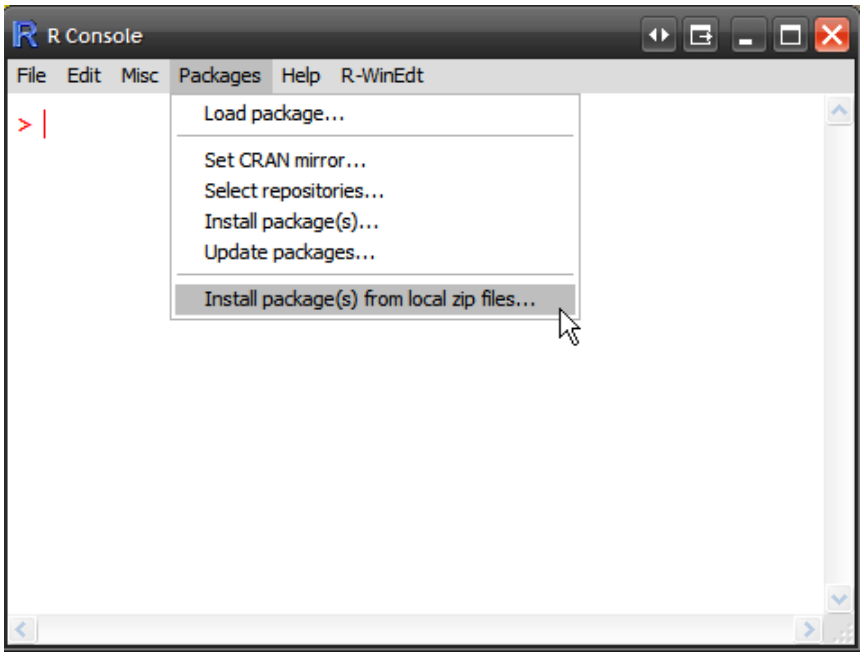
C.1. RMark Installation and First Steps

There are a number of tasks that you need to accomplish prior to using **RMark**. Since you are reading this appendix, chances are good that you will have already installed **MARK** on your computer. If not, refer to the Foreword of this book.

If you haven't already done so, you must install **R** from the **R** Project website

http://cran.r-project.org/

Select 'Windows95 or later', then select base and finally select **r-v.v.v-win32.exe** (where **v.v.v** is the current version (e.g., **R-2.6.1-win32.exe**)). Select run and then follow the directions and choose the default setup by clicking on "next" at each prompt. Download and save the **RMark** package (**RMark.zip**) from **www.phidot.org**. Start **R** from either the desktop icon or from the Start/All Programs list. From within **R**, select **Packages** from the menu and then choose **Install package(s) from local zip** at the bottom of the menu list.



Doing so will show a "select files" window. Navigate to the location where you saved the **RMark.zip** and select the zip file. This will load the package into **c:\Program Files\R\R-v.v.v\library**, where **v.v.v** represents the current version of **R** that you are using (e.g., 2.6.1). Note that **R** installs each version into separate sub-directories of **c:\Program Files\R**. Any updates for **RMark** can be installed as described above over previous versions. If you update **R** versions, you need to repeat the **RMark** installation.

You only need to install **RMark** once but to use **RMark** you will need to issue the command **library(RMark)** in **R** to attach the package every time you start **R**. **R** should respond by displaying the version number that you have installed (e.g. 1.7.3 as shown below) and it will give details about the build date and time and the version of **R** that was used to build the package:

```
This is RMark 1.7.3 Built: R 2.6.1; i386-pc-mingw32; 2007-12-20 09:57:44; windows
```

Most of the time the version of **R** that you are using can be newer than the **R** version used to build **RMark**. However, there are exceptions and if you are having problems with error messages that you cannot resolve, check that the version numbers agree. To avoid manually entering the command each time you initiate **R**, you have a couple of options. You can edit and enter the **library(RMark)** command into the file named "**RProfile.site**" with any text editor. It is located in the directory **C:\Program Files\R\R-v.v.v\etc** where **v.v.v** represents the **R** version. If you add the library(RMark) command to rprofile.site, the **RMark** package will be loaded anytime you start **R**. For additional material on **RProfile.site** see C.24.

Alternatively, you can write a function **.First=function()library(RMark)** and save it into any **.Rdata** workspace from which you will do **MARK** analyses. The **.First()** function is run anytime that particular **.Rdata** workspace is opened. See section C.13 and **R** documentation for more on writing functions.

If you did not select the default location for **MARK** in the installation process (C:\Program Files\Mark) where **RMark** will expect it, then you need to set a variable **MarkPath** to point to the location of the **mark.exe** file. This is best to do in either the **RProfile.site** file or **.First** function. As an example, if you installed **MARK** to **d:\myfiles\mymark**, then in **RProfile.site**, then add the command **MarkPath="d:/myfiles/mymark/"** or **MarkPath="d:\\myfiles\\mymark\\"** (note: Windows uses a backslash ("\") to separate sub-directories in a path but in **R** they are either represented by either a double backslash ("\\") or the simpler single forward slash ("/")). The default value is **MarkPath="C:/Program Files/Mark/"**. Another useful variable that can be set is **MarkViewer**. By default this is set to **notepad.exe** but you can set it to any program like **wordpad.exe** or any text editor you prefer. You need to specify the full directory specification and program name unless the directory is in the **PATH** environment variable.

MARK creates several files when it runs a model (**.inp**, **.out**, **.vcv**, **.res**) and **RMark** retains and uses these four files in the directory where they were created. Everything else **RMark** creates is contained in the **.Rdata** file which is the R workspace. Thus, it is best to create a sub-directory for each set of data you are going to analyze with **RMark**. After you have created the sub-directory copy an empty **.Rdata** file into the new sub-directory and then you can initiate an **R** session by simply double-clicking the **.Rdata** file and any files **RMark** creates will be contained within the subdirectory. It is typically best to start with an empty .Rdata workspace. After a fresh install of **R**, the file **C:\Program Files\R\R-v.v.v\.Rdata** is empty and can be copied to start with an empty workspace. Or all of the workspace contents can be deleted using the command **rm(list=ls(all=TRUE))** or use '**remove all objects**' under the '**Misc**' item in the **R** menu. This is particularly important if you want to mimic some of the examples described in this appendix.

Beyond this appendix, there is an extensive amount of documentation written for each function contained in **RMark**. You can see this documentation using **?mark** within **R** after **library(RMark)** or to see the entire help file double-click on the file

C:\Program Files\R\R-v.v.v\library\RMark \chtml\RMark.chm

where **v.v.v** is the **R** version that you are using. From there you can view or print the entire contents (currently 144 pages).

C.2. A simple example (return of the dippers)

Let's start with a very simple example to explain some of the basic aspects of using **RMark**. Create an empty directory and copy a **.Rdata** file to it. Double click the **.Rdata** file to initiate **R** with that workspace. If there are any objects in the workspace (use **ls()** to see the contents) remove any objects the '**remove all objects**' under the '**Misc**' menu item. Type **library(Rmark)** to attach the package if you have not setup **R** such that the **RMark** package is always attached.

For your first example, we will use the well-known dipper data set that accompanies **MARK** (the dipper data, and all of the other example data files referred to in the **MARK** book and this appendix are found at `http://www.phidot.org/software/mark/docs/book/`. In the drop-down menu '**Book chapters & data files**', select '**Example data files**'). In fact, the dipper data set and a number of others are already contained in the **RMark** package and they can be accessed with the **data** function which extracts the dataframe from the library and puts a copy into your workspace. In addition, with each example set of data, there is some example code for **RMark** to demonstrate use of that particular model. You can run the example code by typing **example(dipper)**, but for this tutorial we will take a simple example and enter each command. If you type data(dipper) and then type **ls()**, it should only show **dipper** as the contents of your workspace.

```
> data(dipper)
> ls()
[1] "dipper"
```

(Note: The object **dipper** is a *dataframe* which is equivalent to a table in MS-ACCESS). Let's get a summary of **dipper** and display the first 5 records to see that it is in the correct format for **RMark**:

```
> summary(dipper)
      ch                 sex
 Length:294         Female:153
 Class :character   Male  :141
 Mode  :character
> dipper[1:5,]
    ch    sex
1 0000001 Female
2 0000001 Female
3 0000001 Female
4 0000001 Female
5 0000001 Female
```

From the above you see that **dipper** has a field named "**ch**" which is a character string containing the capture (encounter) history and it has a field called "**sex**" which is a factor variable. We can tell that "**sex**" is a factor variable because summary shows the frequency of the levels of the factor variable. If it was a numeric variable, summary would show the min, mean, max etc.

We can run a very simple analysis with the **mark** function and assign it to the object "**myexample**" as follows:

```
>myexample=mark(dipper)
```

The output on the screen will be:

```
Output summary for CJS model
Name : Phi(~1)p(~1)

Npar :  2
-2lnL:  666.8377
AICc :  670.866

Beta
                  estimate        se       lcl       ucl
Phi:(Intercept) 0.2421484 0.1020127 0.0422035 0.4420933
p:(Intercept)   2.2262658 0.3251093 1.5890516 2.8634801

Real Parameter Phi

         1        2        3        4        5        6
1 0.560243 0.560243 0.560243 0.560243 0.560243 0.560243
2          0.560243 0.560243 0.560243 0.560243 0.560243
3                   0.560243 0.560243 0.560243 0.560243
4                            0.560243 0.560243 0.560243
5                                     0.560243 0.560243
6                                              0.560243

Real Parameter p

          2         3         4         5         6         7
1 0.9025835 0.9025835 0.9025835 0.9025835 0.9025835 0.9025835
2           0.9025835 0.9025835 0.9025835 0.9025835 0.9025835
3                     0.9025835 0.9025835 0.9025835 0.9025835
4                               0.9025835 0.9025835 0.9025835
5                                         0.9025835 0.9025835
6                                                   0.9025835
```

So what happened and what did that do? First of all let's dissect the command. The piece of code **mark(dipper)** called the function **mark** with the data file **dipper** and used it for the value of its first argument which is called **data**. The equal sign (or <– can be used) was used to assign the result of the **mark** function to the object **myexample** which is stored in the **.Rdata** workspace (although only in memory until the workspace is saved to disk).

So what actually happened inside of the **mark** function? It constructed and ran an analysis using the dataframe **dipper** and the default values for the function arguments **model("CJS")** and **model.parameters** which for this model is to construct **Phi(.)p(.)** (i.e., $\{\varphi.p.\}$) in **MARK** notation and **Phi(~1)p(~1)** in **R** notation. It also used the default values for other function arguments such as **time.intervals** and assumed that there was no group structure for the analysis. Numerous steps were involved but all you need now is the abbreviated version.

The function **mark** examined the capture history (ch) to determine the number of occasions, developed all the necessary structure that it needed, created an **.inp** file for **MARK**, ran **mark.exe** in the background and extracted relevant parts of the **MARK** output files that it needed to create a list of

results. If you use the **R** function **list.files()** you'll see that your directory now contains 4 more files which are the input and the 3 output files from **mark.exe** and each with the prefix **mark001**.

```
> list.files()
[1] "mark001.inp" "mark001.out" "mark001.res" "mark001.vcv"
```

The file **mark001.inp** is the file that would be equivalent to what you would see if you used "Save Structure" rather than directly running the model in the **MARK** interface. **mark001.out** is the text output file from **MARK** with all the results. **mark001.res** is the file of residuals (not currently used by **RMark**) and **mark001.vcv** is a binary file containing the variance-covariance matrices and parameter estimates. All of these files are "linked" to the result object in **R** by the base filename. In this case **myexample** is linked to "**mark001**". Files are numbered sequentially for each analysis with the first available number, but more on that later.

The results from **MARK** were put into the object **myexample** which is a list. If you don't understand the concept of a list in **R**, refer to the **R** tutorial in section C.24. The list created by **RMark** is a slightly special list because it has been assigned to a *class* which means that **R** will treat it differently based on its class. The differential treatment occurs when generic functions like **print** and **summary** are called with the object. You can see the class of an object with the **class** function as follows:

```
> class(myexample)
[1] "mark" "CJS"
```

It has 2 classes with the first being **mark** and the second being the type of mark-recapture model which is "CJS" (Cormack-Jolly-Seber) by default. You need to know this only to understand that when you use functions like **print** and **summary** that **R** actually calls print.mark and **summary.mark**. When **MARK** was finished with the analysis it called **summary** (**summary.mark**) which created the output on the screen. You can see the output again on the screen by simply typing **summary(myexample)**. If you want to save those results to a file, you can use cut and paste to the clipboard or you can use the **sink** function to save any screen output to a file. Use **sink(myfilename)** before issuing the command that generates the output and use any valid file specification in place of **myfilename**. To restore output to the screen use **sink()**.

Let's discuss the summary output to learn some more about **RMark**. The first part of the summary describes the type of model and some basic information. This simple analysis was for the CJS type of analysis and the model name defaults to the concatenation of the formulas used for each of the parameters in the model which was simply **Phi(~1)p(~1)**. The symbol ~ is used to begin a formula when the dependent variable on the left is not specified and implied. The "**1**" represents an intercept so "~ 1" is a model with only the intercept which is equivalent to the 'dot' in **MARK** notation. We will explain much more about specifying formulas later.

After the model description, **summary** provides the number of parameters in the model, the $-2\log(\mathcal{L})$ value and the AIC_c value for the model. We'll see later that the contents of this portion can vary depending on options for parameter counting and use of $\hat{c}$. Next the estimates, standard errors and confidence intervals for the β's are listed as they are shown similarly in the **MARK** output. The estimates are labeled with the type of parameter (e.g., **Phi** or **p** for CJS) and additional names related to the variables in the formula. For this model they are labeled intercept but later we'll see more informative labels. All of the labels for β and real parameters are done automatically and not manually by the user.

The real parameters are shown next in PIM format. For the CJS model, each of the parameters uses an upper-right triangular format for the PIM. The real values are shown for each parameter type (e.g., **Phi** and **p** in this case) and if there were groups defined, the values would be shown by group with a group label. The rows of the triangular PIMS are labeled with the cohort value (time of cohort release) and

the columns are labeled with either the beginning time for time-interval parameters like **Phi** (survival from time 1 to 2 is labeled with 1) or with the time for the occasion for occasion-specific parameters like **p** (re-capture probability at time i is labeled with i). This labeling is controlled by the value given to the beginning time of the experiment (**begin.time**) and by the lengths of the time intervals between occasions (**time.intervals**). For our simple example, we used the default of **begin.time=1** and all the time intervals being 1 so the rows are all labeled from 1 to 6 and the columns are labeled 1 to 6 for **Phi** to represent survival intervals $1 \rightarrow 2, 2 \rightarrow 3, \cdots, 6 \rightarrow 7$ and for **p** the columns are labeled 2 to 7 for the recapture occasions. Had we set **begin.time** to 1990, the rows and columns for **Phi** would have been labeled 1990 to 1995 and the columns for **p** would have been labeled 1991 to 1996.

By showing the real parameters in PIM format it can become readily obvious how the model is parameterized. Although this example is not a particularly good one, it is clear that the constant model was used as all of the real parameters are the same. We'll see more informative examples later.

A summary is nice and later we'll see other types of summaries but how do you look at the whole output file like you do in **MARK**? All you have to do is type the name of the object containing the results (e.g. **myexample**) and hit **enter**. **R** looks for a print method for the object when you type the name of the object (this is discussed in the **R** primer at the end of this appendix). When you type the name of an object with class **mark**, it will use the function **print.mark** to display the object. The function **print.mark** uses the Windows program **notepad.exe** to display the complete output file from **mark.exe**. Until you close the viewer you cannot continue in the **R** session that issued the call to the viewer. If you want to use a different program for viewing output files, simply assign the file specification for the program as a character string to the object **MarkViewer**.

Had we not assigned the results of **mark(dipper)** to **myexample**, **R** would have called **print.mark** to view the output file, and once it was closed, the summary output would have been displayed but no object would have been saved in the **R** workspace. However, the input and 3 output files would still be in the directory but they would not be linked to an object in the **R** workspace. If you were to make this mistake, you can create a **MARK** object in the **R** workspace and link the existing files to it by using the exact same call to **MARK** but adding the **filename** argument and specifying the base filename for the orphaned files. To see how this works, type **mark(dipper)** again but without assigning it to an object and it will create the files **mark002.***, because it is the second analysis that you have run. Then enter the following:

```
> myexample2=mark(dipper,filename="mark002")
```

The code will respond with a query when it sees that the files already exist.

```
Create MARK model with existing file (Y/N)?y
```

By entering "**y**" it will rebuild the model object and link the files to **myexample2**.

Occasionally you will run models and even create an **R** object for them but later decide to delete the **R** objects in the workspace. Deleting the **R** object will not delete the linked files. The function **cleanup** will purge orphaned input and output files. By typing **?cleanup** you will see the help file that describes this function. By typing **cleanup(ask=FALSE)**, all the orphaned files will be deleted. If you want to selectively delete the files, use **cleanup()** and you will be asked to confirm each file deletion.

To see how this works, remove **myexample2**, list the files, use **cleanup** and then list the files again as shown below:

```
> rm(myexample2)
> list.files()
[1] "mark001.inp" "mark001.out" "mark001.res" "mark001.vcv" "mark002.inp"
[6] "mark002.out" "mark002.res" "mark002.vcv"
> cleanup(ask=FALSE)
> list.files()
[1] "mark001.inp" "mark001.out" "mark001.res" "mark001.vcv"
```

C.3. How RMark works

So now that you know how to create, summarize and print a simple model, let's learn more about how **RMark** works so you can fully understand the more realistic examples. To build and run the simple model for the dipper data you did not have to create PIMS nor a design matrix as you might in **MARK**, because **RMark** did it for you. But you may be saying to yourself, that is not really any different than the ability of the **MARK** interface to create pre-specified models. In some ways that is true but with a big difference. The **RMark** package widens the concept of pre-specified models to include user-defined formulas for model definition rather than the limited list of formulas in the **MARK** interface.

So how does it do that? Well with a few tricks and the **R** function **model.matrix**, it is surprisingly simple. The first trick is to realize that your options for developing models are limited by the PIM structure you choose and to fit completely general models without restrictions you need to use what **MARK** calls the *all-different* PIM structure. An all-different PIM is the default PIM type used in **RMark** (although there are some situations where it is useful to specify a simpler PIM structure - see section C.11). You can see the PIM structure by using the PIMS function for **Phi** and **p** with **myexample** as follows:

```
> PIMS(myexample,"Phi",simplified=FALSE)

group = Group 1
   1  2  3  4  5  6
1  1  2  3  4  5  6
2     7  8  9 10 11
3       12 13 14 15
4          16 17 18
5             19 20
6                21

> PIMS(myexample,"p",simplified=FALSE)

group = Group 1
   2  3  4  5  6  7
1 22 23 24 25 26 27
2    28 29 30 31 32
3       33 34 35 36
4          37 38 39
5             40 41
6                42
```

Each of the 21 real parameters in **Phi** (φ) and another 21 real parameters in **p** are given their own unique index, thus the term 'all-different'.

The second trick is realizing that you can *automatically* create and assign "*design data*" to the real parameters based on the model and group structure. This is truly the crux of **RMark** and what makes it possible to use formulae to create models. We use the term "design data" to represent "data" about the model structure, or design. The design data that are created depends on the type of model (e.g, CJS, Multistrata) and the group structure. For a CJS model without groups, the "design data" are occasion (time), age and cohort-specific data. Separate design data are defined for each parameter (e.g., p and φ for CJS models) to allow flexibility and differences in the way design data are handled for each parameter. Also, for labeling it is better to keep them separate since some parameters like **Phi** represent an interval and others like **p** are for an occasion.

Using our first simple example let's describe the design data for the all-different PIMS shown above. There are many different kinds of design data that can be created for any particular example, but there are always several kinds of data that can be created automatically by default. For this example, they are **cohort**, **time** and **age**. We will first describe the design data for **p** which is represented by the indices 22 to 42. Imagine a table of data with 21 rows (one for each parameter) labeled 22 to 42. Let's define a cohort variable that represents the release cohort for each parameter. Rows 22-27 would contain a 1 because they are all for the first cohort, rows 28-32 would contain a 2,. . . , and row 42 would contain a 6. Likewise, if we wanted to create a time variable, then row 22 would contain a 2, rows 23 and 28 would contain a 3,. . . , and rows 27,32,36,39,41, and 42 would contain a 7 because all of those are in the last column for time 7. Likewise we can define a variable we'll call age which is really time-since-marking (TSM) unless all the animals are first released at the same age (e.g., banding young of the year birds). **Age(TSM)** is zero upon first release but it is 1 at the first recapture occasion and age is constant along the diagonals. To create an age variable, the rows 22,28,33,37,40 and 42 in our design data would each have a 1 in the age field, rows 23,29,34,38, and 41 would contain a 2,. . . , and row 27 would contain 6.

We will defer describing how the design data are actually created and can be manipulated but we will show you a summary and list of the first 10 rows of the design data for **p** beginning with index 22 of our design data object, that were created for **myexample** to explain the concept further.

```
group  cohort age   time      Cohort           Age             Time
 1:21   1:6    1:6   2:1   Min.   :0.000   Min.   :1.000   Min.   :0.000
        2:5    2:5   3:2   1st Qu.:0.000   1st Qu.:1.000   1st Qu.:2.000
        3:4    3:4   4:3   Median :1.000   Median :2.000   Median :4.000
        4:3    4:3   5:4   Mean   :1.667   Mean   :2.667   Mean   :3.333
        5:2    5:2   6:5   3rd Qu.:3.000   3rd Qu.:4.000   3rd Qu.:5.000
        6:1    6:1   7:6   Max.   :5.000   Max.   :6.000   Max.   :5.000

    group cohort age time Cohort Age Time
22      1      1   1    2      0   1    0
23      1      1   2    3      0   2    1
24      1      1   3    4      0   3    2
25      1      1   4    5      0   4    3
26      1      1   5    6      0   5    4
27      1      1   6    7      0   6    5
28      1      2   1    3      1   1    1
29      1      2   2    4      1   2    2
30      1      2   3    5      1   3    3
31      1      2   4    6      1   4    4
```

You will likely notice that there are more fields than we described and that some appear to be the same field. First off there is a group field that we didn't describe and it is always 1. This example did not have any group structure, thus all dippers were put in the same group numbered 1. We'll describe the use of grouping variables later. The **cohort**, **age** and **time** fields are created as factor variables as you can notice by the **summary** that shows the counts of the number of entries with each value (level) of the variable. Then there are continuous versions of these variables named **Cohort**, **Age**, and **Time** which have been defined such that they start at 0 for **Cohort** and **Time**. In the **summary** the **min**, **max**, **mean** and **quartiles** are shown for these numeric variables. Capitalization was used to remain consistent with the **MARK** notation (actually, a Colorado State convention) of **p(t)** to represent fitting a model with a separate parameter for each occasion (level of time) and **p(T)** is a continuous trend with an intercept and slope as shown below. In **RMark**, these same models would be ~**time** and ~**Time** respectively.

So far this "trick" may just seem like added complication to the PIM concept. However, that is not the case once you know about the **R** function **model.matrix** which creates design matrices from a formula and data. Now that we have created "design data" for the real parameters, we only need to specify a formula using those data to create the design matrix. While you will never use **model.matrix** directly with the **RMark** package, it is useful to see a demonstration of it to understand how **RMark** works. It is also a useful way to check to make sure your model formula is correct. On the next page, we'll create the design matrix for the first 10 rows (representing parameters 22-31) for the following models for **p**: ~**time**, ~**Time**, ~**Time + age**:

```
model.matrix(~time,myexample$design.data$p[1:10,])

   (Intercept) time3 time4 time5 time6 time7
1            1     0     0     0     0     0
2            1     1     0     0     0     0
3            1     0     1     0     0     0
4            1     0     0     1     0     0
5            1     0     0     0     1     0
6            1     0     0     0     0     1
7            1     1     0     0     0     0
8            1     0     1     0     0     0
9            1     0     0     1     0     0
10           1     0     0     0     1     0

> model.matrix(~Time,myexample$design.data$p[1:10,])

   (Intercept) Time
1            1    0
2            1    1
3            1    2
4            1    3
5            1    4
6            1    5
7            1    1
8            1    2
9            1    3
10           1    4

> model.matrix(~Time+age,myexample$design.data$p[1:10,])
```

```
   (Intercept) Time age2 age3 age4 age5 age6
1            1    0    0    0    0    0    0
2            1    1    1    0    0    0    0
3            1    2    0    1    0    0    0
4            1    3    0    0    1    0    0
5            1    4    0    0    0    1    0
6            1    5    0    0    0    0    1
7            1    1    0    0    0    0    0
8            1    2    1    0    0    0    0
9            1    3    0    1    0    0    0
10           1    4    0    0    1    0    0
```

Once the design data are defined, the **R** function does all the work of creating the design matrix for any formula using the design data. The design matrix is created using the convention called treatment contrasts. That means the first level is used as the intercept and the parameters for the remaining levels are an additive amount relative to the intercept. Also note that **model.matrix** automatically provides all of the label names for the β parameters as the column names of the design matrix.

If you only had these automatic design data, **RMark** would be fairly useful but would still be less than optimal. Later we'll show how you can manipulate and extend the design data to make it completely general and much more useful for designing models beyond these basic cookie-cutter types.

While all-different PIMS are necessary to enable creation of any model, they become problematic when the size of the problem is such that the number of real parameters (number of rows in the design matrix) exceeds 5000. For some data sets and models this happens easily. Some large models will not run in **mark.exe** due to insufficient memory when the variance-covariance matrix for the real parameters is created. However, even if **MARK** can run the model it is quite inefficient and slow to use a design matrix with say 5000 real parameters and only 2 columns for the **Phi(.)p(.)** model.

This difficulty led to 'trick number 3' which is the concept of simplifying the design matrix. If you have the **Phi(.)p(.)** model with 5000 real parameters, 2500 of the design matrix rows would have a 1 in column 1 and a 0 in column 2 and the other 2500 rows would have a 0 in column 1 and a 1 in column 2. That is quite redundant and really all one needs is the 2 unique rows to convey the information in the 5000 rows. After the design matrix is created with the all-different PIMS, **RMark** simplifies it to contain only the unique rows and re-codes the PIMS. A link is maintained between the original indices and the new simplified indices. Simplification has important consequences for the viability of the modeling approach in **RMark** and the speed at which **mark.exe** completes the analysis.

To see the simplified and recoded PIMS for a model, you can use the **PIMS** function but this time using the default value of **simplified=TRUE**. If you use it with **myexample** as below you'll see that the 42 parameters have been recoded to the 2 unique parameters.

```
> PIMS(myexample,"Phi")

group = Group 1
   1  2  3  4  5  6
1  1  1  1  1  1  1
2     1  1  1  1  1
3        1  1  1  1
4           1  1  1
5              1  1
```

```
6                1

> PIMS(myexample,"p")

group = Group 1
   2  3  4  5  6  7
1  2  2  2  2  2  2
2     2  2  2  2  2
3        2  2  2  2
4        2  2  2  2
5              2  2
6                 2
```

If you can simplify and recode the PIMS to the unique values, why would you want to keep the links to the original all-different indices? Because the original all-different PIMS provides a compatible foundation for all the analyses of the same data set using the same underlying type of model (e.g., CJS). With the all-different PIMS it is easier to display real parameters in PIM format, associate labels to the real parameters and to use model averaging on the real parameters from different models which will have different simplified PIM coding.

It should be helpful to examine the recoded PIMS for some other models, so without describing how we got them, we show the recoded PIMS for parameter **p** with ~**time**, ~**Time** and ~**Time+age** models with **Phi**(~**1**) as shown above for design matrices:

```
~time or ~Time group = Group 1
   2  3  4  5  6  7
1  2  3  4  5  6  7
2     3  4  5  6  7
3        4  5  6  7
4           5  6  7
5              6  7
6                 7


~Time + age group = Group 1
   2  3  4  5  6  7
1  2  3  4  5  6  7
2     8  9 10 11 12
3       13 14 15 16
4          17 18 19
5             20 21
6                22
```

Notice that the recoded PIMS for the ~**Time+age** model has 21 different parameters as with the all-different PIMS because with that model all of the rows of the design matrix for p are different. However, the PIM is recoded to start at 2 because **Phi(**~1**)** only requires a single parameter.

To a large extent the PIM/design simplification is transparent to you as a user in analyzing the data except that simplification does create a conflict between the labeling of real parameters in the **MARK** output and the labeling of real parameters in output from **summary** and other functions in **R**. When the

PIMS are simplified there is no attempt to create a unique meaningful label for the real parameters in the input file sent to **mark.exe**. It uses the label associated with the first real parameter translated to the new PIM coding. However, the labeling of real parameters in **R** is maintained with the use of the all-different PIM structure. So use **R** when you want to look at real parameter values with their labels and ignore the labels in the **MARK** output file for real parameters.

PIM simplification is done for all parameters except for parameters that use the **mlogit** links like ψ in the multistrata model and **pent** in **POPAN**. The **mlogit** link assures that the sum of a specified set of probabilities sums to 1 but it is implemented in **MARK** by using a sum of the unique real parameters indices and not the full set of real parameters. So for example, if you had 5 strata (**A** to **E**) and you wanted to estimate 4 real parameters for transitions from **A** by constraining equality for **D** and **E** ($\psi^{AB}, \psi^{AC}, \psi^{AD} = \psi^{AE}$). If you give these 4 parameters indices 1 to 4, then the **mlogit** link will work properly because it will sum across all 4, but if you give the parameters the indices 1,2,3,3 to constrain the last two parameters then the sum will be only the first 3 parameters and it will not sum the third parameter twice. Thus, an all-different PIM structure is required for parameters that use the **mlogit** link and any equality constraints must be implemented with the design matrix without any simplification of the PIMS. This restriction on **mlogit** links does not affect how you use **RMark** but may affect the speed at which **MARK** computes the parameter estimates because the number of parameters and the size of the design matrix is larger without PIM simplification.

As we showed above, **model.matrix** in **R** is the workhorse for creation of design matrices from formula; however, it cannot directly cope with individual covariates in the design matrix structure of **MARK** which uses the name of the individual covariate in the design matrix. To be generally useful, the formula notation needed to encompass individual covariates and this led to 'trick number 4' which is probably the only clever trick in the **RMark** implementation. But we'll delay divulging it until section C.16.

There are just a few things more you should understand before we move on. Note that the indices are "stacked on top of each other" to get unique indices for all of the parameters. Thus, for our example there are 21 φ parameters numbered 1 to 21 and 21 p parameters numbered 22 to 42. This ordering of the index numbers is done in a consistent fashion for each model. For example, p always follows φ in the CJS model. However, in most places in the code where you have to specify indices (see C.11 - fixing real parameters) it will typically only need to identify the parameter with the parameter-specific index which is the row number in the design matrix. Thus, in most cases for p, the parameters are identified by the indices 1 to 21. The only exception is situations in which you are referring to parameter indices across parameter types (e.g., both φ and p) as with the function **covariate.predictions** (C.16).

For most models in **MARK**, the design matrix could be displayed in the following manner:

design for parameter 1	0	0	0
0	*design for parameter 2*	0	0
0	0	$\ddots$	0
0	0	0	*design for parameter* ***k***

where none of the different types of parameters (e.g., p, φ etc) share columns of the design matrix. Parameter types can share the same covariate (e.g., $\varphi_t p_t$), but the effect of that covariate is not the same for the different types of parameters so the covariates are represented by different columns in the design

matrix. For most models, this works quite well but there are some exceptions including parameters “p" and “c" in the closed and robust design models, parameters “p1" and “p2" in the **MSOccupancy** model, and “GammaPrime" and “GammaDoublePrime" in the robust design models. In each of these cases the parameter has a different name but it is effectively the same type of parameter, so it is quite reasonable to build models in which they “share" covariates or are equated. To accommodate this exception, the parameter listed first is set as the dominant parameter and the formula for the dominant parameter is given a special argument “**share**" that can be set to **TRUE** or **FALSE**. If it is set to **TRUE**, then the design data are combined ‘on the fly’ and an extra column is added for the non-dominant parameter to enable fitting additive models. See section C.19 for an example.

C.4. Dissecting the function “mark"

Now that you have been introduced to some of the ideas on the inner workings of **RMark** like design data and PIM structure and simplification, we’ll discuss the steps that are taken in producing an analysis and along the way we will expand the concept of design data to include group structure. The function **mark** is actually quite simple because it is a convenience function that calls 5 other functions that actually do the work in the following order:

1. **process.data**
2. **make.design.data**
3. **make.mark.model**
4. **run.mark.model**
5. **summary.mark**

Why do you care? Primarily because the function has dual calling modes for efficiency and to enable adding/modifying the design data. Depending on the arguments that you pass **mark**, it will either start with **process.data** or it will skip directly to **make.mark.model**. This allows you to do the first 2 steps once, optionally modify the design data, and then run a whole series of models on the data without repeating the first 2 steps in each call to **mark**.

C.4.1. Function process.data

The first function **process.data** literally does what its name implies. It takes the input data frame and the user-defined arguments and creates a list (processed data) containing the data and numerous defined attributes that the remaining functions use in defining the analysis models. The following are the primary attributes that are set:

1. **model**: the type of analysis model (e.g., "CJS", "Known", "POPAN"); see help for function **mark (?mark)** for a complete listing of the supported models
2. **begin.time**: the time of the first capture/release occasion for labeling
3. **time.intervals**: the lengths of the time intervals between capture occasions
4. **groups**: the list of factor variables in the data to define groups
5. **initial.ages**: the age of animals at first capture/release corresponding to the levels of the age grouping variable (**age.var**)

6. **nocc**: number of capture/encounter occasions which is determined from the contents of the "**ch**" field in the data and the type of analysis **model(model)**.

As an example, we will use the dipper data and the field **sex** to create 2 groups in the data and define fictitious beginning time and time intervals for the data:

```
> data(dipper)
> dipper.process=process.data(dipper,model="CJS",begin.time=1980,
                              time.intervals=c(1,.5,1,.75,.25,1),groups="sex")
```

The resulting object (**dipper.process**) is a list containing the data and its attributes. The names of the elements of the list can be viewed with the **names** function:

```
> names(dipper.process)
 [1] "data"             "model"            "mixtures"         "freq"
 [5] "nocc"             "nocc.secondary"   "time.intervals"   "begin.time"
 [9] "age.unit"         "initial.ages"     "group.covariates" "nstrata"
[13] "strata.labels"
```

Note that there are many more attributes than described above. Some – like **mixtures**, **nstrata**, **nocc.secondary** and **strata.labels** – are only relevant to specific models but these are often included with a default, NULL or empty value for models in which they are not relevant. Specific elements of the list can be extracted as illustrated:

```
> dipper.process$nocc
[1] 7
> dipper.process$group.covariates
     sex
1 Female 2   Male
> dipper.process$begin.time
[1] 1980
> dipper.process$strata.labels
character(0)
> dipper.process$nocc.secondary
NULL
> dipper.process$time.intervals
[1] 1.00 0.50 1.00 0.75 0.25 1.00
```

From the first 5 rows of the field **freq** it is obvious that this is the structure used to create the frequency data for the **MARK** input file with the defined grouping structure and the column labels as the group labels:

```
> dipper.process$freq[1:10,]
   sexFemale sexMale
1          1       0
2          1       0
3          1       0
4          1       0
5          1       0
```

The structure of the encounter history and the analysis depends on the analysis model that you choose like "**CJS**" above. Thus, it is necessary to process the data frame (data) containing the encounter history and a chosen model to define the relevant values which will be used by the remaining functions. For example, number of capture occasions (**nocc**) is automatically computed based on the length of the encounter history (**ch**) in data; however, this is dependent on the type of analysis model. For models such as "**CJS**", "**Pradel**" and others, it is simply the length of **ch**. Whereas, for "**Burnham**" and "**Barker**" models, the encounter history contains capture and resight/recovery values so **nocc** is one-half the length of **ch**. Likewise, the number of **time.intervals** depends on the model. For models, such as "**CJS**", "**Pradel**" and others, the number of **time.intervals** is **nocc**-1; whereas, for capture-recovery (or resight) models the number of **time.intervals** is **nocc**. The default time interval is unit time (1) and if this is adequate, the function will assign the appropriate length; otherwise the appropriate number of values must be given.

A processed data frame can only be analyzed using the **model** that was specified in the call to **process.data**. The **model** value is used by the functions **make.design.data** and **make.mark.model** to define the design data and the appropriate input file structure for **MARK**. Thus, if the data are going to be analyzed with different underlying models, create different processed data objects possibly using the type of **model** as an extension. For example,

```
dipper.cjs=process.data(dipper,model="CJS")
dipper.popan=process.data(dipper,model="POPAN")
```

The **process.data** function will report any inconsistencies in the lengths of the capture history values and when invalid entries are given in the capture history. For example, with the "**CJS**" model, the capture history should only contain 0 and 1 whereas for "**Barker**" it can contain 0,1,2. For "**Multistrata**" models, the code will automatically identify the number of strata (**nstrata**) and strata labels (**strata.labels**) based on the unique alphabetic codes used in the capture histories. For "**Robust**" design models, the number of secondary occasions (**nocc.secondary**) is determined by the specified **time.intervals**.

The argument **begin.time** specifies the time for the first capture/release occasion. This is used in creating the levels of the **time** factor variable in the design data and for labeling parameters. If **begin.time** varies by group, enter a vector of times with one for each group.

The argument groups can contain one or more character strings specifying the names of factor variables contained in **data**. A group is created for each unique combination of the levels of the factor variables. Further examples of grouping and use of age variables will be given later and they can be found in the help documentation with **R** (**?process.data** and **?example.data**).

C.4.2. Function make.design.data

The next step is to create the design data and PIM structure which depends on the selected type of analysis model (e.g., CJS or Multistrata), number of occasions, grouping variables and other attributes of the data that were defined in the processed data, which is the first and primary argument to the function **make.design.data** that creates the design data. For parameters with triangular PIMS the default design data are **cohort**, **age** and **time** and any grouping factor variables that were defined. For parameters with square PIMS, there is only one row so the cohort variable is not automatically included in the design data but there are ways to create a cohort structure in this case with groups.

In creating the factor variables for cohort, age, and time, a separate factor level is created for each value of the variable. However, you can optionally bin the values into intervals in creating the factor variable. For example, if birds were always classified as either young (< 1) or as adult (1+), then

age.bins could be specified in the call to **make.design.data**. However, if you wanted the option to model age based on all levels of the factor and other models with some ages collapsed into intervals then it is best to allow **make.design.data** to create the default factor variables and create additional design data with the function **add.design.data** or using **R** statements and functions. There are many other features of **make.design.data** including restricting parameters to use "time" or "constant" PIMS, setting the subtraction stratum for "Multistrata" models, and automatic removal of unused design data. These features are described in the help files (**?make.design.data** and **?add.design.data**) and they are described in more detail in later sections.

For now, a simple example with the dipper data will suffice to illustrate this step and explain the basic concepts. But before we do that we'll reprocess the data to use annual time intervals rather than the fictitious ones used above:

```
> dipper.process=process.data(dipper,model="CJS",begin.time=1980,groups="sex")
```

The result of a call to **make.design.data** is a list of design data, so one naming convention is to use **ddl** (design data list) as the suffix and the data name as the prefix as follows:

```
> dipper.ddl=make.design.data(dipper.process)
```

Before we look at the design data, let's run a simple model with **mark** but this time rather than specifying the data file, we'll specify the processed data and the design data list. When **MARK** is called with these 2 arguments it recognizes that they have already been created and skips to step 3 to create and run the model directly.

```
> myexample2=mark(dipper.process,dipper.ddl)

Output summary for CJS model Name : Phi(~1)p(~1)

Npar :  2
-2lnL:  666.8377
AICc :  670.866

Beta
                 estimate        se       lcl       ucl
Phi:(Intercept) 0.2421484 0.1020127 0.0422035 0.4420933
p:(Intercept)   2.2262658 0.3251093 1.5890517 2.8634800


Real Parameter Phi Group:sexFemale
      1980     1981     1982     1983     1984     1985
1980 0.560243 0.560243 0.560243 0.560243 0.560243 0.560243
1981          0.560243 0.560243 0.560243 0.560243 0.560243
1982                   0.560243 0.560243 0.560243 0.560243
1983                            0.560243 0.560243 0.560243
1984                                     0.560243 0.560243
1985                                              0.560243

Group:sexMale
         1980     1981     1982     1983     1984     1985
```

```
1980 0.560243 0.560243 0.560243 0.560243 0.560243 0.560243
1981          0.560243 0.560243 0.560243 0.560243 0.560243
1982                   0.560243 0.560243 0.560243 0.560243
1983                            0.560243 0.560243 0.560243
1984                                     0.560243 0.560243
1985                                              0.560243


Real Parameter p Group:sexFemale
          1981      1982      1983      1984      1985      1986
1980 0.9025835 0.9025835 0.9025835 0.9025835 0.9025835 0.9025835
1981           0.9025835 0.9025835 0.9025835 0.9025835 0.9025835
1982                     0.9025835 0.9025835 0.9025835 0.9025835
1983                               0.9025835 0.9025835 0.9025835
1984                                         0.9025835 0.9025835
1985                                                   0.9025835

Group:sexMale
          1981      1982      1983      1984      1985      1986
1980 0.9025835 0.9025835 0.9025835 0.9025835 0.9025835 0.9025835
1981           0.9025835 0.9025835 0.9025835 0.9025835 0.9025835
1982                     0.9025835 0.9025835 0.9025835 0.9025835
1983                               0.9025835 0.9025835 0.9025835
1984                                         0.9025835 0.9025835
1985                                                   0.9025835
```

If you are following along with these commands and did not get the results above make sure that you reprocessed the data with the annual intervals and then created the design data before entering the call to **mark** because the results will vary with different time intervals. Notice that the results are exactly the same as the first analysis we did with the dipper data; however, the real parameter summaries are displayed for each sex because it was used to define groups.

Now let's look at the non-simplified PIMS for φ and compare them to the design data that were created.

```
> PIMS(myexample2,"Phi",simplified=FALSE)

group = sexFemale
     1980 1981 1982 1983 1984 1985
1980    1    2    3    4    5    6
1981         7    8    9   10   11
1982             12   13   14   15
1983                  16   17   18
1984                       19   20
1985                            21

group = sexMale
     1980 1981 1982 1983 1984 1985
1980   22   23   24   25   26   27
1981        28   29   30   31   32
```

```
1982             33   34   35   36
1983                  37   38   39
1984                       40   41
1985                            42
```

To accommodate the group structure 42 possible real parameter indices were created for **Phi** with 1-21 for females and 22-42 for males. The same structure was also created for **p**. If we look at the names of the design data list

```
> names(dipper.ddl)
[1] "Phi"      "p"        "pimtypes"
```

we see that there are 3 elements in the list. The first 2 are the design data for the parameters in the CJS model (**Phi** and **p**) and the last is a list of the type of PIMS used which in this case is the default of all-different. We can examine the design data for **Phi** as follows (with abbreviated output):

```
> dipper.ddl$Phi
    group cohort age time Cohort Age Time    sex
1  Female   1980   0 1980      0   0    0 Female
2  Female   1980   1 1981      0   1    1 Female
3  Female   1980   2 1982      0   2    2 Female
4  Female   1980   3 1983      0   3    3 Female
5  Female   1980   4 1984      0   4    4 Female
6  Female   1980   5 1985      0   5    5 Female
7  Female   1981   0 1981      1   0    1 Female
8  Female   1981   1 1982      1   1    2 Female
9  Female   1981   2 1983      1   2    3 Female
10 Female   1981   3 1984      1   3    4 Female
<...>
22   Male   1980   0 1980      0   0    0   Male
23   Male   1980   1 1981      0   1    1   Male
37   Male   1983   0 1983      3   0    3   Male
38   Male   1983   1 1984      3   1    4   Male
39   Male   1983   2 1985      3   2    5   Male
40   Male  1984    0 1984      4   0    4   Male
41   Male   1984   1 1985      4   1    5   Male
42   Male   1985   0 1985      5   0    5   Male
```

Rows (indices) 1 and 22 have the same design data except that row 1 is for females and row 22 is for males. Any grouping variables are automatically included into the design data. A field "group" is created which is a unique combination of the different values of each grouping variable and then a separate field is included for each grouping variable. With a single grouping variable, like sex, the 2 fields are identical. The pre-defined models in **MARK** like $\{\varphi_{g*t}\}$ are equivalent to using the field group in the formula. As we will show later, the inclusion of each grouping variable allows additive models to be created with the grouping variables rather than just using the group field which is the full interaction of the grouping variables.

C.5. More simple examples

Hopefully you now have a basic understanding of how PIMS, design data and design matrices are created in **RMark** and we can move on to learning how to specify formula for analysis models. Along the way we'll reiterate and expand on the material we have presented so far. We will continue on with the dipper data with **sex** used for groups to describe using **formula** with the existing design data created by default and then we'll consider examples that work with user-defined supplemental design data.

Following along the lines of **MARK**, a model is described by sub-models for each parameter of the particular type of mark-recapture analysis. With the dipper data we have been using the CJS model with parameters φ and p, and so far we have been using the default model which is a constant value for each parameter. A parameter specification (sub-model) is defined by a list, although in most circumstances the list will only contain a single element named the **formula**. For reasons that will be obvious later, the parameter specifications should be assigned to an object named with a prefix being the parameter name and the suffix being a description for the formula or some other strategy like numbering. For example, with the simple model we have constructed so far the parameter specifications would be:

```
> Phi.dot=list(formula=~1)
> p.dot=list(formula=~1)
```

The parameter specifications are used with the **mark** argument **model.parameters** to define the model. The default model we ran earlier could also be specified as:

```
> myexample2=mark(dipper.process,dipper.ddl,model.parameters=list(Phi=Phi.dot,p=p.dot))
```

Now the parameter specification **Phi.dot** and **p.dot** are identical so you could have done the following:

```
> myexample2=mark(dipper.process,dipper.ddl,model.parameters=list(Phi=Phi.dot,p=Phi.dot))
```

and gotten the same results but that could be a bit confusing and later you'll see that there are advantages to having a separate parameter specification object for each parameter even if they have the same values.

So, let's create some more parameter specifications solely for demonstration purposes as some of these models may not make sense for the dipper data:

```
Phi.time=list(formula=~time)
Phi.sex=list(formula=~sex)
Phi.sexplusage=list(formula=~sex+age)
p.time=list(formula=~time)
p.Time=list(formula=~Time)
p.Timeplussex=list(formula=~Time+sex)
```

By including the dot models, we could easily specify 16 (4×4) different models for all the combinations of these parameter specifications. Hmm, how do we name all these models to keep them straight? One way is to use the data name and add on the parameter specifications as in the following examples:

```
dipper.phi.dot.p.dot=
          mark(dipper.process,dipper.ddl,model.parameters=list(Phi=Phi.dot,p=p.dot))
dipper.phi.time.p.dot=
          mark(dipper.process,dipper.ddl,model.parameters=list(Phi=Phi.time,p=p.dot))
```

```
dipper.phi.sex.p.dot=
          mark(dipper.process,dipper.ddl,model.parameters=list(Phi=Phi.sex,p=p.dot))
dipper.phi.sex.p.Timeplussex=
      mark(dipper.process,dipper.ddl,model.parameters=list(Phi=Phi.sex,p=p.Timeplussex))
dipper.phi.time.p.time=
          mark(dipper.process,dipper.ddl,model.parameters=list(Phi=Phi.time,p=p.time))
dipper.phi.sexplusage.p.dot=
      mark(dipper.process,dipper.ddl,model.parameters=list(Phi=Phi.sexplusage,p=p.dot))
```

See how easy it is? No messing with PIMS or design matrices. You are certainly getting the idea but let's look at the last 3 models in more detail to learn some more. If you ran these by simply copying the text into **R**, the output will have passed by on the screen but we can simply repeat it with the **summary** function:

```
> summary(dipper.phi.sex.p.Timeplussex)

Output summary for CJS model
Name : Phi(~sex)p(~Time + sex)

Npar :  5
-2lnL:  664.1672
AICc :  674.3101

Beta
                 estimate        se        lcl       ucl
Phi:(Intercept) 0.1947163 0.1403108 -0.0802928 0.4697254
Phi:sexMale     0.7547928 0.1989333 -0.3351165 0.4447022
p:(Intercept)   1.2297543 0.6455548 -0.0355331 2.4950417
p:Time          0.3162690 0.2255297 -0.1257693 0.7583073
p:sexMale       0.4290287 0.6660079 -0.8763468 1.7344042

Real Parameter Phi Group:sexFemale
          1980      1981      1982      1983      1984      1985
1980 0.5485258 0.5485258 0.5485258 0.5485258 0.5485258 0.5485258
1981           0.5485258 0.5485258 0.5485258 0.5485258 0.5485258
1982                     0.5485258 0.5485258 0.5485258 0.5485258
1983                               0.5485258 0.5485258 0.5485258
1984                                         0.5485258 0.5485258
1985                                                   0.5485258

Group:sexMale
          1980      1981      1982      1983      1984      1985
1980 0.5620557 0.5620557 0.5620557 0.5620557 0.5620557 0.5620557
1981           0.5620557 0.5620557 0.5620557 0.5620557 0.5620557
1982                     0.5620557 0.5620557 0.5620557 0.5620557
1983                               0.5620557 0.5620557 0.5620557
1984                                         0.5620557 0.5620557
1985                                                   0.5620557
```

```
Real Parameter p Group:sexFemale
          1981      1982      1983      1984      1985      1986
1980 0.7737756 0.8243386 0.8655639 0.8983077 0.9237786 0.9432727
1981           0.8243386 0.8655639 0.8983077 0.9237786 0.9432727
1982                     0.8655639 0.8983077 0.9237786 0.9432727
1983                               0.8983077 0.9237786 0.9432727
1984                                         0.9237786 0.9432727
1985                                                   0.9432727

Group:sexMale
          1981      1982      1983      1984      1985      1986
1980 0.8400746 0.8781527 0.9081557 0.9313485 0.9490134 0.9623168
1981           0.8781527 0.9081557 0.9313485 0.9490134 0.9623168
1982                     0.9081557 0.9313485 0.9490134 0.9623168
1983                               0.9313485 0.9490134 0.9623168
1984                                         0.9490134 0.9623168
1985                                                   0.9623168
```

Let's look at the **mark** result object some more so you can see how to extract various parts of the results. We see that the names of the elements are:

```
> names(dipper.phi.sex.p.Timeplussex)
 [1] "data"             "model"            "title"            "model.name"
 [5] "links"            "mixtures"         "call"             "parameters"
 [9] "time.intervals"   "number.of.groups" "group.labels"     "nocc"
[13] "begin.time"       "covariates"       "fixed"            "design.matrix"
[17] "pims"             "design.data"      "strata.labels"    "mlogit.list"
[21] "simplify"         "model.parameters" "results"          "output"
```

The field **output** is the link to the input and output files

```
> dipper.phi.sex.p.Timeplussex$output
[1] "mark012"
```

The value may be different for you depending on how many models you have run and whether you removed models and used the **cleanup** function. The element **pims** is the all-different PIMS for the model but the extractor function **PIMS** produces clearer output than simply typing the command **dipper.phi.sex.p.Timeplussex$pims**. The element **model.parameters** is simply the value of the **mark** argument with the same name; whereas, the **parameters** field is for internal use with various attributes set for each parameter. Likewise, the links between the simplified PIMS and the non-simplified PIMS contained in the list element **simplify** is only useful internally. The design matrix for the simplified model structure is also contained in the result as a matrix:

```
> dipper.phi.sex.p.Timeplussex$design.matrix
                        Phi:(Intercept) Phi:sexMale p:(Intercept) p:Time p:sexMale
Phi gFemale c1980 a0 t1980 "1"             "0"         "0"           "0"    "0"
Phi gMale c1980 a0 t1980   "1"             "1"         "0"           "0"    "0"
p gFemale c1980 a1 t1981   "0"             "0"         "1"           "0"    "0"
p gFemale c1980 a2 t1982   "0"             "0"         "1"           "1"    "0"
```

```
p gFemale c1980 a3 t1983   "0"          "0"        "1"          "2"    "0"
p gFemale c1980 a4 t1984   "0"          "0"        "1"          "3"    "0"
p gFemale c1980 a5 t1985   "0"          "0"        "1"          "4"    "0"
p gFemale c1980 a6 t1986   "0"          "0"        "1"          "5"    "0"
p gMale c1980 a1 t1981     "0"          "0"        "1"          "0"    "1"
p gMale c1980 a2 t1982     "0"          "0"        "1"          "1"    "1"
p gMale c1980 a3 t1983     "0"          "0"        "1"          "2"    "1"
p gMale c1980 a4 t1984     "0"          "0"        "1"          "3"    "1"
p gMale c1980 a5 t1985     "0"          "0"        "1"          "4"    "1"
p gMale c1980 a6 t1986     "0"          "0"        "1"          "5"    "1"
```

The list element of most interest is **`results`**, a list containing extracted values from the **MARK** output files:

```
> names(dipper.phi.sex.p.Timeplussex$results)
 [1] "lnl"              "deviance"         "npar"             "n"
 [5] "AICc"             "beta"             "real"             "beta.vcv"
 [9] "derived"          "derived.vcv"      "covariate.values" "singular"
```

The definitions of the elements are as follows:

- **`lnl`**: $-2\log\mathcal{L}$ Likelihood value
- **`deviance`**: difference between null deviance and model deviance
- **`npar`**: Number of parameters (always the number of columns in design matrix)
- n: effective sample size
- **`AICc`**: Small sample corrected AIC using **`npar`**
- **`beta`**: data frame of β parameters with estimate, standard error (**`se`**), lower confidence limit (**`lcl`**), and upper confidence limit (**`ucl`**)
- **`real`**: data frame of unique (simplified) real parameters with estimate, standard error (**`se`**), lower confidence limit (**`lcl`**), and upper confidence limit (**`ucl`**), and notation for fixed parameters
- **`beta.vcv`**: variance-covariance matrix for β
- **`derived`**: dataframe of derived parameters if any
- **`derived.vcv`**: variance-covariance matrix for derived parameters if any
- **`covariate.values`**: dataframe with fields **`Variable`** and **`Value`** which are the covariate names and value used for real parameter estimates in the **MARK** output
- **`singular`**: indices of β parameters that are non-estimable or at a boundary

The individual elements can be extracted using list notation. For example, the data frame of the β parameters:

```
> dipper.phi.sex.p.Timeplussex$results$beta
                 estimate        se        lcl       ucl
Phi:(Intercept) 0.1947163 0.1403108 -0.0802928 0.4697254
Phi:sexMale     0.7547928 0.1989333 -0.3351165 0.4447022
p:(Intercept)   1.2297543 0.6455548 -0.0355331 2.4950417
p:Time          0.3162690 0.2255297 -0.1257693 0.7583073
p:sexMale       0.4290287 0.6660079 -0.8763468 1.7344042
```

or the data frame of the unique (simplified) real parameters:

```
> dipper.phi.sex.p.Timeplussex$results$real
                             estimate        se       lcl       ucl fixed
Phi gFemale c1980 a0 t1980 0.5485258 0.0347473 0.4799376 0.6153188
Phi gMale c1980 a0 t1980   0.5620557 0.0349965 0.4927116 0.6290571
p gFemale c1980 a1 t1981   0.7737756 0.1130024 0.4911177 0.9237935
p gFemale c1980 a2 t1982   0.8243386 0.0721225 0.6387191 0.9256861
p gFemale c1980 a3 t1983   0.8655639 0.0495235 0.7365526 0.9368173
p gFemale c1980 a4 t1984   0.8983077 0.0424479 0.7803679 0.9564497
p gFemale c1980 a5 t1985   0.9237786 0.0418005 0.7910491 0.9748739
p gFemale c1980 a6 t1986   0.9432727 0.0411246 0.7866317 0.9868417
p gMale c1980 a1 t1981     0.8400746 0.0970652 0.5603827 0.9558434
p gMale c1980 a2 t1982     0.8781527 0.0627026 0.6956116 0.9578565
p gMale c1980 a3 t1983     0.9081557 0.0430622 0.7823503 0.9645393
p gMale c1980 a4 t1984     0.9313485 0.0345158 0.8248454 0.9750509
p gMale c1980 a5 t1985     0.9490134 0.0312841 0.8397867 0.9850955
p gMale c1980 a6 t1986     0.9623168 0.0291539 0.8408254 0.9919649
```

Remember that the labels for the real parameters in the simplified model can be misleading due to the simplification process. To view all of the real parameters with standard errors, use **summary** as follows (the output has been abbreviated):

```
> summary(dipper.phi.sex.p.Timeplussex,se=T)

Output summary for CJS model Name : Phi(~sex)p(~Time + sex)

Real Parameter p
                         par.index  estimate        se       lcl       ucl fixed
p gFemale c1980 a1 t1981         3 0.7737756 0.1130024 0.4911177 0.9237935
p gFemale c1980 a2 t1982         4 0.8243386 0.0721225 0.6387191 0.9256861
p gFemale c1980 a3 t1983         5 0.8655639 0.0495235 0.7365526 0.9368173
p gFemale c1980 a4 t1984         6 0.8983077 0.0424479 0.7803679 0.9564497
p gFemale c1980 a5 t1985         7 0.9237786 0.0418005 0.7910491 0.9748739
p gFemale c1980 a6 t1986         8 0.9432727 0.0411246 0.7866317 0.9868417
p gFemale c1981 a1 t1982         4 0.8243386 0.0721225 0.6387191 0.9256861
p gFemale c1981 a2 t1983         5 0.8655639 0.0495235 0.7365526 0.9368173
p gFemale c1981 a3 t1984         6 0.8983077 0.0424479 0.7803679 0.9564497
p gFemale c1981 a4 t1985         7 0.9237786 0.0418005 0.7910491 0.9748739
p gFemale c1981 a5 t1986         8 0.9432727 0.0411246 0.7866317 0.9868417
p gFemale c1982 a1 t1983         5 0.8655639 0.0495235 0.7365526 0.9368173
p gFemale c1982 a2 t1984         6 0.8983077 0.0424479 0.7803679 0.9564497
p gFemale c1982 a3 t1985         7 0.9237786 0.0418005 0.7910491 0.9748739
p gFemale c1982 a4 t1986         8 0.9432727 0.0411246 0.7866317 0.9868417
p gFemale c1983 a1 t1984         6 0.8983077 0.0424479 0.7803679 0.9564497
p gFemale c1983 a2 t1985         7 0.9237786 0.0418005 0.7910491 0.9748739
p gFemale c1983 a3 t1986         8 0.9432727 0.0411246 0.7866317 0.9868417
p gFemale c1984 a1 t1985         7 0.9237786 0.0418005 0.7910491 0.9748739
p gFemale c1984 a2 t1986         8 0.9432727 0.0411246 0.7866317 0.9868417
```

```
p gFemale c1985 a1 t1986         8 0.9432727 0.0411246 0.7866317 0.9868417
p gMale   c1980 a1 t1981         9 0.8400746 0.0970652 0.5603827 0.9558434
p gMale   c1980 a2 t1982        10 0.8781527 0.0627026 0.6956116 0.9578565
p gMale   c1980 a3 t1983        11 0.9081557 0.0430622 0.7823503 0.9645393
p gMale   c1980 a4 t1984        12 0.9313485 0.0345158 0.8248454 0.9750509
```

The **par.index** field is the index within the simplified set of real parameters (i.e., the recoded parameter index). The label for the real parameter uses a short hand notation in which **g** is for group, **c** for cohort, **a** for age and **t** for time. After each letter is the value of the variable. In other types of mark-recapture models, like **Multistrata**, additional values are added like s for stratum, and **t** for **to stratum**, for movement from one stratum to another stratum.

C.6. Design covariates in RMark

There are 2 types of covariates used in **RMark**. You have already seen examples of the first type which is the design covariate (design data). Design covariates are linked to the parameters in the model and specify differences in the parameters associated with the model structure (e.g., time, cohort) or with group structure of the animals (e.g., sex) because different parameters are used for different groups of animals. The second type of covariate is individual covariates which specify differences in the individual animals. The distinction between the types is not entirely clear-cut because design covariates for group structure are individual covariates because each animal has its own value. However, group design covariates have 2 important restrictions: 1) they must be a factor variable which means they will typically have a small number of unique values (e.g., sex=M or F), and 2) the value cannot change over time. Thus, individual covariates are typically used for numeric variables (e.g., mass, length) or for covariates where the value changes over time (e.g., trap dependence). You can code factor variables as individual covariates by creating $k-1$ dummy variables (0/1) for a factor variable with k levels, but it is usually better to use factor variables as group design covariates. Design covariates are stored in the design data (`ddl`) and individual covariates remain with the encounter history data. Use of individual covariates in data and models is described in C.16 and in this section we demonstrate how the flexibility of design covariates can be used to expand the usefulness of model formula.

So far, all of the examples we have created have only used the design data created by default using the group and model structure. While that may be all that is needed in many instances, additional design data can be created and used in formula and this substantially adds to the flexibility of model development. What kinds of design data can be added and why would you want to do that? Any data that are relevant to the model and group structure can be added to the design data. These can be dummy variables that enable "effects" to be modeled for subsets of any of the design data fields. For example, below we will create a design data field called **Flood** for the dipper example which is 1 in years with floods and 0 in non-flood years. Dummy variables are equivalent to coding a column in the design matrix as you do with the standard **MARK** interface. Or the added design data fields may create a factor variable with new intervals of existing design data. For example, we'll create a design data field that bins ages as young (0 and 1) and sub-adult (2-3), and adult (4^+) (*note*: this and other treatments of the dipper data may not be realistic for dippers). Finally, the added design data could be a numeric field that is specific to some parameter. For example, we'll create an effort field for each sampling occasion in the dipper data to model capture probability.

Design data can be created and modified with any relevant **R** statement or function. We will start with a simple example using the dipper data using the fictitious dates we assigned. With the dipper data, between sampling occasions 1981-1982 and 1982-1983 there were severe floods that could have reduced survival in those periods and capture probability may have differed in 1982 (note: use of these

dates may not reflect the true situation). To model this effect, we will define a **Flood** variable that is 1 for flood periods and 0 otherwise. Remember that there are different design data for each parameter, so a **Flood** field has to be defined for each parameter that will use the field in the model. Because the timing of the effect varies for φ and p, the definitions of those variables are different.

```
> dipper.ddl$Phi$Flood=0
> dipper.ddl$Phi$Flood[dipper.ddl$Phi$time==1981 |  dipper.ddl$Phi$time==1982]=1
> dipper.ddl$p$Flood=0
> dipper.ddl$p$Flood[dipper.ddl$p$time==1982]=1
```

The first statement above creates a **Flood** field for **Phi** and assigns the value 0 to all values. The second statement assigns 1 to **Flood** for those rows in the dataframe for which time is either 1981 (for interval 1981 to 1982) or 1982 (for interval 1982 to 1983). The last 2 statements define the **Flood** variable for capture probability. Once the data have been created they can be used in models as shown below:

```
> Phi.Flood=list(formula=~Flood)
> p.Flood=list(formula=~Flood)
> dipper.phi.flood.p.dot=
    mark(dipper.process,dipper.ddl,model.parameters=list(Phi=Phi.Flood,p=p.dot))
> dipper.phi.flood.p.flood=
    mark(dipper.process,dipper.ddl,model.parameters=list(Phi=Phi.Flood,p=p.Flood))
```

While you can use any **R** statement to create design data, in many instances the design data you are creating is a modification of existing data or merges new data with existing data, so some functions were created to simplify the process. If the new design data are simply creating bins (intervals) of time, age, or cohort, then you can use the function **add.design.data**. For example, if we want to create age intervals for survival (young, sub-adult, and adult) as we described above, we can do it as follows:

```
> dipper.ddl=add.design.data(dipper.process, dipper.ddl,
   parameter="Phi", type="age", bins=c(0,1,3,6),name="ageclass")
```

If we summarize the design data for **Phi**, we see that the variable we chose to name **ageclass** has been defined properly:

```
> summary(dipper.ddl$Phi)
    group     cohort   age      time        Cohort          Age
 Female:21   1980:12   0:12   1980: 2   Min.   :0.000   Min.   :0.000
 Male  :21   1981:10   1:10   1981: 4   1st Qu.:0.000   1st Qu.:0.000
             1982: 8   2: 8   1982: 6   Median :1.000   Median :1.000
             1983: 6   3: 6   1983: 8   Mean   :1.667   Mean   :1.667
             1984: 4   4: 4   1984:10   3rd Qu.:3.000   3rd Qu.:3.000
             1985: 2   5: 2   1985:12   Max.   :5.000   Max.   :5.000
     Time           sex         Flood         ageclass
 Min.   :0.000   Female:21   Min.   :0.0000   [0,1]:22
 1st Qu.:2.000   Male  :21   1st Qu.:0.0000   (1,3]:14
 Median :4.000               Median :0.0000   (3,6]: 6
 Mean   :3.333               Mean   :0.2381
 3rd Qu.:5.000               3rd Qu.:0.0000
 Max.   :5.000               Max.   :1.0000
```

It is always a good idea to examine the design data after you have created it to make sure that the intervals were defined as expected and that they included the entire range of the data. In the definition of **ageclass**, a "(" means the interval is open on the left which means that value is not included in the interval. Whereas a square bracket ("[" or "]") is for a closed interval which means the interval end point is included. If we decided that the intervals should be shifted to the left, the easiest way is as follows:

```
> dipper.ddl=add.design.data(dipper.process, dipper.ddl,
    parameter="Phi", type="age",
       bins=c(0,1,3,6),name="ageclass",right=FALSE,replace=TRUE)
```

Had we not used **replace=T**, we would have gotten the following error:

```
Error in add.design.data(dipper.process, dipper.ddl, parameter =
"Phi",  : Variable  ageclass  already in design data. Use
replace=TRUE if you want to replace current values
```

Now **ageclass** defines the intervals 0,1 to 2, and 3+ for modeling age effects in **Phi**:

```
> summary(dipper.ddl$Phi$ageclass)
[0,1) [1,3) [3,6]
   12    18    12
```

Had we issued the following function call:

```
> dipper.ddl=add.design.data(dipper.process, dipper.ddl,
    parameter="Phi", type="age",bins=c(1,3,6),name="badageclass")
```

then when we summarized the field, the presence of **NA**s make it apparent that the defined bins did not span the range of the **age** field:

```
> summary(dipper.ddl$Phi$badageclass)
[1,3] (3,6]  NA's
   24     6    12
```

The NAs occurred in this case because 0 was excluded. Notice that the intervals are always closed on the far left and far right. Since we do not want this field, by assigning **NULL** to the field

```
> dipper.ddl$Phi$badageclass=NULL
```

it is removed from the design data for Phi:

```
> names(dipper.ddl$Phi)
 [1] "group"    "cohort"   "age"    "time"  "Cohort"   "Age"    "Time"    "sex"
 [8] "Flood"    "ageclass"
```

In many situations the additional design data are simply covariates to be used in place of occasion/-time effects. Examples are effort, weather, or observers which vary for occasions and may be useful to simplify modeling of capture probability rather than time-varying parameters. For this situation, the function **merge_design.covariates** was created. The following is an example in which fictitious effort data were created for the dipper data:

```
> df=data.frame(time=c(1980:1986),effort=c(10,5,2,8,1,2,3))
> dipper.ddl$p=merge_design.covariates(dipper.ddl$p,Xdf)
> summary(dipper.ddl$p$effort)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  1.000   2.000   2.000   3.095   3.000   8.000
```

So why is the maximum value for effort only 8 and not 10? For the CJS model there is no capture probability for 1980, so the value is ignored. The function is less forgiving if you forget to include data for one of the times:

```
> df=data.frame(time=c(1980:1985),effort=c(10,5,2,8,1,2))
> dipper.ddl$p=merge_design.covariates(dipper.ddl$p,df)

  Error in merge_design.covariates(dipper.ddl$p,df) :

  df does not contain a time value for each time in design data
```

The dataframe can contain any number of covariates with any valid names and the only restriction is that it must contain a field named **time** with values that match those in the design data. The dataframe can be created as above or functions like **read.table** can be used to import data in a file into a dataframe. For more details on these functions, refer to the **R** help files on **read.table** and **data.frame**.

Let's create another model that uses those new design data.

```
> Phi.ageclass.plus.sex=list(formula=~ageclass+sex)
> p.effort.plus.sex=list(formula=~effort+sex)
> dipper.phi.ageclassplussex.p.effortplussex =
    mark(dipper.process,dipper.ddl,model.parameters=list(Phi=
       Phi.ageclass.plus.sex,p= p.effort.plus.sex))

Output summary for CJS model
Name : Phi(~ageclass + sex)p(~effort + sex)

Npar :  7
-2lnL:  665.1266
AICc :  679.3946

Beta
                    estimate        se        lcl       ucl
Phi:(Intercept)    0.1964602 0.1750104 -0.1465602 0.5394805
Phi:ageclass[1,3)  0.1033126 0.2258833 -0.3394186 0.5460439
Phi:ageclass[3,6] -0.1287800 0.4376419 -0.9865582 0.7289982
Phi:sexMale        0.0306891 0.2046965 -0.3705160 0.4318943
p:(Intercept)      2.3193847 0.5699104  1.2023604 3.4364090
p:effort          -0.0901470 0.1098187 -0.3053917 0.1250977
p:sexMale          0.4862940 0.6626337 -0.8124681 1.7850561

Real Parameter Phi Group:sexFemale
          1980      1981      1982      1983      1984      1985
1980 0.5489577 0.5743870 0.5743870 0.5169136 0.5169136 0.5169136
```

```
1981           0.5489577 0.5743870 0.5743870 0.5169136 0.5169136
1982                     0.5489577 0.5743870 0.5743870 0.5169136
1983                               0.5489577 0.5743870 0.5743870
1984                                         0.5489577 0.5743870
1985                                                   0.5489577

Group:sexMale
          1980      1981      1982      1983      1984      1985
1980 0.5565444 0.5818718 0.5818718 0.5245725 0.5245725 0.5245725
1981           0.5565444 0.5818718 0.5818718 0.5245725 0.5245725
1982                     0.5565444 0.5818718 0.5818718 0.5245725
1983                               0.5565444 0.5818718 0.5818718
1984                                         0.5565444 0.5818718
1985                                                   0.5565444
```

Notice the diagonal patterns in **Phi** as it relates to the final **ageclass** definition that we used.

It is also possible to assign group- and time-specific covariates to the design data with the **merge_design.covariates** function. The following is an example using the dipper data in which effort is sex (group) specific. A dataframe (df) is constructed with the group and time-specific effort values and this is then used in the call to **merge_design.covariates**. See the help file for that function for more details.

```
> df=data.frame(group=c(rep("Female",7),rep("Male",7)),
            time=rep(c(1980:1986),2),effort=c(10,5,2,8,1,2,3,20,10,4,16,2,4,6))
> df
     group time effort
  1  Female 1980     10
  2  Female 1981      5
  3  Female 1982      2
  4  Female 1983      8
  5  Female 1984      1
  6  Female 1985      2
  7  Female 1986      3
  8    Male 1980     20
  9    Male 1981     10
  10   Male 1982      4
  11   Male 1983     16
  12   Male 1984      2
  13   Male 1985      4
  14   Male 1986      6

> dipper.process=process.data(dipper,group="sex",begin.time=1980)
> dipper.ddl=make.design.data(dipper.process)
> dipper.ddl=merge_design.covariates(dipper.ddl$p,df,bygroup=TRUE)

> dipper.ddl$p

     group cohort age time Cohort Age Time    sex effort
  1  Female   1980   1 1981      0   1    0 Female      5
  2  Female   1980   2 1982      0   2    1 Female      2
  3  Female   1980   3 1983      0   3    2 Female      8
  <...>
  29   Male   1981   2 1983      1   2    2   Male     16
```

```
30   Male   1981   3 1984      1   3    3   Male      2
31   Male   1981   4 1985      1   4    4   Male      4
32   Male   1981   5 1986      1   5    5   Male      6
<...>
40   Male   1984   1 1985      4   1    4   Male      4
41   Male   1984   2 1986      4   2    5   Male      6
42   Male   1985   1 1986      5   1    5   Male      6
```

C.7. Comparing results from multiple models

We have put together quite a few models with lots of different names! So how do we keep track of the models and how do we summarize them for model selection and possible model averaging of parameter estimates? Later we will explain more organized approaches but they all tie back to the functions we will use now. The first function is **collect.models** which collects all of the models that have been run into a single list and it calls the function **model.table**, although the latter can be called separately. Although **collect.models** does have arguments in most cases it will be called without arguments and assigned to a list that you can name to help you remember its contents:

```
> dipper.cjs.results=collect.models()
```

What did this do? It looked through all of the objects in the workspace and collected any object that had a class of "**mark**". If the workspace included more than one type of **MARK** model, like "CJS" and "POPAN", it would have issued a warning message. Although it does not matter for this session, the collection of objects can be limited to a particular type of model as follows:

```
> dipper.cjs.results=collect.models(type="CJS")
```

Like the models which have a class of "**mark**" the list resulting from **collect.models** has a class of "**marklist**" and some of the generic functions treat it differently. For example, the **print** function provides a listing of the **model.table** element of the list rather than printing each list element which are the various model results.

```
> dipper.cjs.results
                                  model npar     AICc DeltaAICc       weight Deviance
3                     Phi(~Flood)p(~1)    3 666.1597   0.00000 0.5767865187 77.62566
4                   Phi(~Flood)p(~Flood)    4 668.1557   1.99605 0.2126073874 77.58357
2                           Phi(~1)p(~1)    2 670.8660   4.70638 0.7548324523 84.36055
11                          Phi(~1)p(~1)    2 670.8660   4.70638 0.7548324523 58.15788
12                          Phi(~1)p(~1)    2 670.8660   4.70638 0.7548324523 84.36055
5                         Phi(~sex)p(~1)    3 672.7331   6.57343 0.0215582207 84.19909
9                        Phi(~time)p(~1)    7 673.9980   7.83838 0.0114533494 77.25297
6                 Phi(~sex)p(~Time + sex)    5 674.3101   8.15044 0.0097987244 81.69012
7                    Phi(~sex + age)p(~1)    8 678.9925  12.83286 0.0009427448 80.17008
8                 Phi(~sex + age)p(~Time)    9 679.1198  12.96015 0.0008846140 78.21000
1  Phi(~ageclass + sex)p(~effort + sex)    7 679.3946  13.23491 0.0007710649 82.64951
10                   Phi(~time)p(~time)   11 679.5879  13.42824 0.0007000190 74.47310
```

The table of model results is fashioned along the lines of the results table shown in the **MARK** interface. By default the table is displayed in ascending order for AIC_c. The number on the left hand-side of the table is the order of the model in the list. If we look at the names of the list elements we see

that the first 12 are the names of the models that we created and the last is the **model.table** which is the dataframe that is displayed above.

```
> names(dipper.cjs.results)
 [1] "dipper.phi.ageclassplussex.p.effortplussex"
 [2] "dipper.phi.dot.p.dot"
 [3] "dipper.phi.flood.p.dot"
 [4] "dipper.phi.flood.p.flood"
 [5] "dipper.phi.sex.p.dot"
 [6] "dipper.phi.sex.p.Timeplussex"
 [7] "dipper.phi.sexplusage.p.dot"
 [8] "dipper.phi.sexplusage.p.Time"
 [9] "dipper.phi.time.p.dot"
[10] "dipper.phi.time.p.time"
[11] "myexample"
[12] "myexample2"
[13] "model.table"
```

The model with the lowest AIC_c is the third model in the list. Notice that models 2, 11 and 12 are all the same model. That is because the collection includes the first examples we created named **myexample** and **myexample2**. We certainly don't want models duplicated in the list and especially if we use model averaging. There are different ways they can be removed from the list. One approach would be to use the **rm** function in **R** to remove them from the workspace and then recreate the list. The more direct approach would be to use the function **remove.mark** to remove models 11 and 12 as follows:

```
> dipper.cjs.results=remove.mark(dipper.cjs.results,c(11,12))
> dipper.cjs.results
                              model npar     AICc DeltaAICc       weight Deviance
3                  Phi(~Flood)p(~1)    3 666.1597   0.00000 0.6478308242 77.62566
4              Phi(~Flood)p(~Flood)    4 668.1557   1.99605 0.2387947959 77.58357
2                      Phi(~1)p(~1)    2 670.8660   4.70638 0.0615863089 84.36055
5                    Phi(~sex)p(~1)    3 672.7331   6.57343 0.0242136031 84.19909
9                   Phi(~time)p(~1)    7 673.9980   7.83838 0.0128640885 77.25297
6            Phi(~sex)p(~Time + sex)    5 674.3101   8.15044 0.0110056589 81.69012
7               Phi(~sex + age)p(~1)    8 678.9925  12.83286 0.0010588651 80.17008
8            Phi(~sex + age)p(~Time)    9 679.1198  12.96015 0.0009935742 78.21000
1Phi(~ageclass + sex)p(~effort + sex)    7 679.3946  13.23491 0.0008660389 82.64951
10                Phi(~time)p(~time)   11 679.5879  13.42824 0.0007862422 74.47310
```

Each of the 10 models is stored in the list and the individual named objects in the workspace are no longer needed. The names of the model objects can be collected with the function **collect.model.names** and easily removed as follows:

```
> rm(list=collect.model.names(ls())) # result of function used as argument to 'rm'
> ls()
 [1] "df"                    "dipper"                "dipper.cjs.results"
 [4] "dipper.ddl"            "dipper.process"        "p.dot"
 [7] "p.effort.plus.sex"     "p.Flood"               "p.time"
[10] "p.Time"                "p.time.fixed"          "p.Timeplussex"
[13] "Phi.ageclass.plus.sex" "Phi.dot"               "Phi.Flood"
[16] "Phi.sex"               "Phi.sex.plus.age"      "Phi.sexplusage"
[19] "Phi.time"
```

The objects defined for the parameter model specifications (e.g., **p.flood**) remain but the model results were removed from the workspace. You can summarize, print, and manipulate any of the models by simply referring to the model as a particular list element (e.g., **summary(dipper.cjs.results[[3]])**). Maintaining the model results in a **marklist** is a much tidier way to organize results of analyses; however, more importantly, model averaging requires the results to be contained in a **marklist**. Also, adjusting model selection for over-dispersion is much easier if the models are maintained in a **marklist**.

C.8. Producing model-averaged parameter estimates

The function **model.average** provides model averaging of the real parameters either for a single type of parameter (e.g., "**Phi**" or "**p**") or for all parameters. No facility is provided for model averaging the beta parameters although all of the values are available in the **marklist** to do so. All of the real parameters can be averaged over the models as follows:

```
> dipper.mod.avg=model.average(dipper.cjs.results,vcv=TRUE)
```

By default, the function returns a dataframe of the model averaged estimates with standard errors but not confidence intervals. If you include **vcv=TRUE**, it will return a list with a dataframe named estimates which includes the estimates with standard errors and confidence intervals and a variance-covariance matrix.

```
> names(dipper.mod.avg)
[1] "estimates" "vcv.real"
```

Model-averaged estimates, standard errors and confidence intervals are provided in the estimates dataframe:

```
> summary(dipper.mod.avg$estimates)
  par.index        estimate            se                lcl              ucl
Min.   : 1.00   Min.   :0.4771   Min.   :0.02991   Min.   :0.3833   Min.   :0.5719
1st Qu.:21.75   1st Qu.:0.6023   1st Qu.:0.03016   1st Qu.:0.5339   1st Qu.:0.6667
Median :42.50   Median :0.7506   Median :0.03357   Median :0.6609   Median :0.8066
Mean   :42.50   Mean   :0.7364   Mean   :0.03439   Mean   :0.6592   Mean   :0.7956
3rd Qu.:63.25   3rd Qu.:0.9000   3rd Qu.:0.03433   3rd Qu.:0.8237   3rd Qu.:0.9454
Max.   :84.00   Max.   :0.9034   Max.   :0.04926   Max.   :0.8241   Max.   :0.9593
```

The field **par.index** is the parameter index for the **all-different** PIM. In this case the first 42 (2 groups of 21) are for **Phi** and the last 42 are for **p**. Unless you need a covariance between parameters of different types, it is more useful to construct the model-averaged estimates by parameter type because the default design data are added to the estimates dataframe which provides some context for the estimates.

```
> dipper.Phi.mod.avg=model.average(dipper.cjs.results,"Phi",vcv=TRUE)
> dipper.Phi.mod.avg$estimates[1:5,]
  par.index  estimate         se       lcl       ucl fixed  group cohort age time Cohort Age Time    sex
1         1 0.6024905 0.03488516 0.5325433 0.6684866       Female   1980   0 1980      0   0    0 Female
2         2 0.4771467 0.04853963 0.3839480 0.5719644       Female   1980   1 1981      0   1    1 Female
3         3 0.4776554 0.04857463 0.3843736 0.5725222       Female   1980   2 1982      0   2    2 Female
4         4 0.6023430 0.03432131 0.5335474 0.6673187       Female   1980   3 1983      0   3    3 Female
5         5 0.6022495 0.03435730 0.5333826 0.6672924       Female   1980   4 1984      0   4    4 Female
```

The estimates, standard errors and variance-covariance matrix are constructed as described by Burnham and Anderson (2002: chapter 4). Confidence intervals for the model-averaged estimates were somewhat more challenging. To provide valid intervals for bounded parameters (e.g., $0 < \varphi < 1$), the model-average variance-covariance matrix of the real parameters are transformed to a variance-covariance matrix for the estimates transformed into the appropriate link space using the Delta-method (see Appendix B). Then asymptotic 95% normal confidence intervals are constructed for the transformed link values and the interval end points are then back-transformed into real parameters. That same method is used to construct confidence intervals for the real parameters for a single model in **MARK**.

C.9. Quasi-likelihood adjustment

An estimate of $\hat{c}$ for over-dispersion can be derived using the **TEST2+TEST3** χ^2/df from program **RELEASE** (see Chapter 5 for full details).

Program **RELEASE** can be run with the function **`release.gof`** as shown below with the dipper data:

```
> data(dipper)
> dipper.processed=process.data(dipper,model="CJS",groups="sex")
> release.gof(dipper.processed)

 RELEASE NORMAL TERMINATION
       Chi.square df      P
 TEST2     7.5342  6 0.2743
 TEST3    10.7735 15 0.7685
 Total    18.3077 21 0.6295
```

If you add the argument **`view=TRUE`**, the **RELEASE** output file named (`Releasennn.out`) will be displayed in a window.

Alternatively $\hat{c}$ can be estimated using the median $\hat{c}$ procedure but this is not currently supported in **RMark**. However, you can export the input file and the output from the global model to **MARK** and use the **MARK** interface to run the median $\hat{c}$ procedure. See section C.21 for a description of exporting to **MARK**.

Adjustments for over-dispersion are implemented with the function **`adjust.chat`** which sets the value of chat for an individual model or all models in a **`marklist`**. For example, we will set the value of $\hat{c}$ to 2 for the set of dipper results we just created:

```
> dipper.cjs.results=adjust.chat(2,dipper.cjs.results)
```

Doing so does nothing more than setting an element called $\hat{c}$ in each model to 2 in this case. It does not adjust standard errors or confidence intervals in any of the model objects but that is done with functions that extract the results (e.g., **`get.real`**). However, it does adjust the **`model.table`** values:

```
>  dipper.cjs.results
                                  model npar    QAICc DeltaQAICc       weight QDeviance chat
3                     Phi(~Flood)p(~1)    3 336.1083   0.000000 0.4661602174  38.81283    2
2                         Phi(~1)p(~1)    2 337.4472   1.338942 0.2386644310  42.18028    2
4                  Phi(~Flood)p(~Flood)    4 338.1254   2.017100 0.1700307784  38.79179    2
5                       Phi(~sex)p(~1)    3 339.3950   3.286715 0.0901226832  42.09955    2
6               Phi(~sex)p(~Time + sex)    5 342.2265   6.118215 0.0218766933  40.84506    2
9                       Phi(~time)p(~1)    7 344.1330   8.024731 0.0084330973  38.62649    2
1  Phi(~ageclass + sex)p(~effort + sex)    7 346.8313  10.722996 0.0021880957  41.32475    2
```

```
7                Phi(~sex + age)p(~1)    8 347.6689  11.560662 0.0014393600  40.08504    2
8              Phi(~sex + age)p(~Time)    9 348.7762  12.667990 0.0008274011  39.10500    2
10                 Phi(~time)p(~time)   11 351.1128  15.004529 0.0002572427  37.23655    2
```

The **model.table** now contains QAIC$_c$ values and the remaining computations based on it instead of AIC$_c$. The ordering of the models is also changed in this case.

C.10. Coping with identifiability

Now let's look at the summary output from the **Phi(~time)p(~time)** model which we know will be over-parameterized because only the product of the last φ and p are estimable:

```
Output summary for CJS model
Name : Phi(~time)p(~time)

Npar :  12  (unadjusted=11)
-2lnL:  656.9502
AICc :  681.7057 (unadjusted=679.58789)

Beta
                   estimate          se          lcl         ucl
Phi:(Intercept)  0.9354557   0.7685213   -0.5708461   2.4417575
Phi:time1981    -1.1982745   0.8706688   -2.9047853   0.5082364
Phi:time1982    -1.0228292   0.8049137   -2.6004601   0.5548017
Phi:time1983    -0.4198589   0.8091476   -2.0057882   1.1660705
Phi:time1984    -0.5360978   0.8031424   -2.1102571   1.0380614
Phi:time1985     0.2481368 244.9012000 -479.7582200 480.2544900
p:(Intercept)    0.8292835   0.7837354   -0.7068380   2.3654050
p:time1982       1.6556230   1.2913788   -0.8754795   4.1867256
p:time1983       1.5220926   1.0729148   -0.5808205   3.6250057
p:time1984       1.3767410   0.9884819   -0.5606835   3.3141654
p:time1985       1.7950894   1.0688773   -0.2999101   3.8900889
p:time1986      -0.0147563 187.0364400 -366.6061900 366.5766800

Real Parameter Phi Group:sexFemale
          1980      1981      1982      1983      1984      1985
1980 0.7181808 0.4346709 0.4781705 0.6261176 0.5985334 0.7655931
1981           0.4346709 0.4781705 0.6261176 0.5985334 0.7655931
1982                     0.4781705 0.6261176 0.5985334 0.7655931
1983                               0.6261176 0.5985334 0.7655931
1984                                         0.5985334 0.7655931
1985                                                   0.7655931

Group:sexMale
          1980      1981      1982      1983      1984      1985
1980 0.7181808 0.4346709 0.4781705 0.6261176 0.5985334 0.7655931
1981           0.4346709 0.4781705 0.6261176 0.5985334 0.7655931
1982                     0.4781705 0.6261176 0.5985334 0.7655931
1983                               0.6261176 0.5985334 0.7655931
```

```
1984                                          0.5985334 0.7655931
1985                                                    0.7655931

Real Parameter p Group:sexFemale
          1981      1982      1983      1984      1985      1986
1980 0.6962034 0.9230769 0.9130435 0.9007892 0.9324138 0.6930734
1981           0.9230769 0.9130435 0.9007892 0.9324138 0.6930734
1982                     0.9130435 0.9007892 0.9324138 0.6930734
1983                               0.9007892 0.9324138 0.6930734
1984                                         0.9324138 0.6930734
1985                                                   0.6930734

Group:sexMale
          1981      1982      1983      1984      1985      1986
1980 0.6962034 0.9230769 0.9130435 0.9007892 0.9324138 0.6930734
1981           0.9230769 0.9130435 0.9007892 0.9324138 0.6930734
1982                     0.9130435 0.9007892 0.9324138 0.6930734
1983                               0.9007892 0.9324138 0.6930734
1984                                         0.9324138 0.6930734
1985                                                   0.6930734
```

Note that the number of parameters is shown as 12 and AIC_c is calculated based on 12, but an unadjusted parameter count and AIC_c are also shown with the proper count of 11. The **mark** function assumes that all parameters are identifiable and if the parameter count in the **MARK** output is less than the number of columns in the design matrix, it adjusts the count and AIC_c value if the default value of the argument **adjust=TRUE** is used. It also keeps the values reported by **MARK** in **results$npar.unadjusted** and **results$AICc.unadjusted** and these are reported in **summary**.

Why not trust the values computed by **MARK**? The ability of **MARK** to count the number of parameters correctly is impaired when using design matrices and it will often not count parameters that are estimable but are at a boundary (0 or 1 for φ or p) which can happen easily with sparse data sets (the technical details of how **MARK** counts parameters are presented in Chapter 4). Overly complex models that have numerous parameters that are at boundaries can appear to be the best model because the parameters are counted improperly. It is more conservative to assume that all parameters are estimable.

When you know that some parameters are not identifiable and should not be counted there are a couple of ways to proceed. One approach is to fix the value of one of the parameters to 1 so it will not be counted and the other parameter is then an estimate of the product of the parameters. This can be done with the argument fixed in the parameter specification list as follows:

```
p.time.fixed=list(formula=~time,fixed=list(time=1986,value=1))
dipper.phi.time.p.time=
    mark(dipper.process,dipper.ddl,model.parameters=list(Phi=Phi.time,p=p.time.fixed))

Output summary for CJS model Name : Phi(~time)p(~time)

Npar :  11 -2lnL:  656.9502 AICc :  679.5879

Beta
                  estimate        se        lcl       ucl
Phi:(Intercept)  0.9354601 0.7685246 -0.5708483 2.4417684
Phi:time1981    -1.1982793 0.8706724 -2.9047973 0.5082387
```

```
Phi:time1982    -1.0228337 0.8049168 -2.6004706 0.5548031
Phi:time1983    -0.4198627 0.8091504 -2.0057975 1.1660720
Phi:time1984    -0.5361021 0.8031460 -2.1102683 1.0380640
Phi:time1985    -0.8128580 0.7947326 -2.3705340 0.7448179
p:(Intercept)    0.8292792 0.7837366 -0.7068447 2.3654031
p:time1982       1.6556296 1.2913815 -0.8754783 4.1867374
p:time1983       1.5220968 1.0729155 -0.5808177 3.6250112
p:time1984       1.3767444 0.9884827 -0.5606817 3.3141704
p:time1985       1.7950930 1.0688789 -0.2999097 3.8900957
p:time1986       0.0000000 0.0000000  0.0000000 0.0000000

Real Parameter Phi Group:sexFemale
          1980      1981      1982      1983      1984      1985
1980 0.7181817 0.4346708 0.4781705 0.6261177 0.5985334 0.5306122
1981           0.4346708 0.4781705 0.6261177 0.5985334 0.5306122
1982                     0.4781705 0.6261177 0.5985334 0.5306122
1983                               0.6261177 0.5985334 0.5306122
1984                                         0.5985334 0.5306122
1985                                                   0.5306122

Real Parameter p Group:sexFemale
          1981      1982      1983      1984      1985 1986
1980 0.6962025 0.9230771 0.9130435 0.9007891 0.9324138    1
1981           0.9230771 0.9130435 0.9007891 0.9324138    1
1982                     0.9130435 0.9007891 0.9324138    1
1983                               0.9007891 0.9324138    1
1984                                         0.9324138    1
1985                                                      1
```

Fixing parameters can get a little tricky with additive models, so an alternative approach is to use **adjust=FALSE** with **mark** to accept the parameter counts from **MARK** or afterward you can use the function **adjust.parameter.count** to change the parameter count to a new value and AIC_c is subsequently recalculated. If you are going to accept the **MARK** parameter counts, make sure they are correct! In complex models with dozens of parameters, it is quite possible that the optimization code does not reach the global maximum and parameters end up at boundaries and are not counted. Indices for the parameters that are not counted by **MARK** are stored in **results$singular**. You should always check these parameters and ascertain whether it is likely that they are at boundaries and whether they are estimable. If you have any doubts, rerun the model with new starting values as we show in the next example.

The final example we ran earlier demonstrates a situation in which a parameter is at a boundary but is properly estimated:

```
> summary(dipper.phi.sexplusage.p.dot)

Output summary for CJS model
Name : Phi(~sex + age)p(~1)

Npar :  8  (unadjusted=7)
-2lnL:  662.6472
AICc :  678.9925 (unadjusted=676.91513)
```

```
Beta
                   estimate           se         lcl         ucl
Phi:(Intercept)   0.1647608 1.696575e-01  -0.1677680   0.4972896
Phi:sexMale       0.0830684 1.995167e-01  -0.3079844   0.4741211
Phi:age1          0.0173059 2.538808e-01  -0.4803006   0.5149123
Phi:age2          0.3599325 3.692076e-01  -0.3637144   1.0835793
Phi:age3         -0.0402832 5.407864e-01  -1.1002246   1.0196581
Phi:age4          0.2645044 8.873705e-01  -1.4747419   2.0037506
Phi:age5        -19.8742890 1.076391e-08 -19.8742890 -19.8742890
p:(Intercept)     2.2565572 3.289010e-01   1.6119113   2.9012031


Real Parameter Phi Group:sexFemale
          1980      1981      1982      1983      1984         1985
1980 0.5410973 0.5453913 0.6282445 0.5310793 0.6056982 2.755882e-09
1981           0.5410973 0.5453913 0.6282445 0.5310793 6.056982e-01
1982                     0.5410973 0.5453913 0.6282445 5.310793e-01
1983                               0.5410973 0.5453913 6.282445e-01
1984                                         0.5410973 5.453913e-01
1985                                                   5.410973e-01
```

```
Group:sexMale
          1980      1981      1982      1983      1984         1985
1980 0.5616421 0.5658982 0.6474300 0.5517010 0.6253534 2.994585e-09
1981           0.5616421 0.5658982 0.6474300 0.5517010 6.253534e-01
1982                     0.5616421 0.5658982 0.6474300 5.517010e-01
1983                               0.5616421 0.5658982 6.474300e-01
1984                                         0.5616421 5.658982e-01
1985                                                   5.616421e-01


Real Parameter p Group:sexFemale
          1981      1982      1983      1984      1985      1986
1980 0.9052146 0.9052146 0.9052146 0.9052146 0.9052146 0.9052146
1981           0.9052146 0.9052146 0.9052146 0.9052146 0.9052146
1982                     0.9052146 0.9052146 0.9052146 0.9052146
1983                               0.9052146 0.9052146 0.9052146
1984                                         0.9052146 0.9052146
1985                                                   0.9052146

Group:sexMale
          1981      1982      1983      1984      1985      1986
1980 0.9052146 0.9052146 0.9052146 0.9052146 0.9052146 0.9052146
1981           0.9052146 0.9052146 0.9052146 0.9052146 0.9052146
1982                     0.9052146 0.9052146 0.9052146 0.9052146
1983                               0.9052146 0.9052146 0.9052146
1984                                         0.9052146 0.9052146
1985                                                   0.9052146
```

```
> dipper.phi.sexplusage.p.dot$results$singular
[1] 7
```

The seventh β for the age 5 φ effect is at a boundary such that survival from age 5 to 6 is estimated to be zero, We can see if this a numerical problem by rerunning the model and changing the initial value for beta 7 using the **initial** argument of **mark** as follows:

```
> initial= dipper.phi.sexplusage.p.dot$results$beta$estimate
> initial[7]=0
> dipper.phi.sexplusage.p.dot
    =mark(dipper.process,dipper.ddl,model.parameters
      =list(Phi=Phi.sexplusage,p=p.dot),initial=initial)
```

Setting the "singular" βs to zero and refitting the model will often help the optimization move away from the boundary and find the global maximum. That is the approach that is taken if you set the argument **retry** in **mark** to a non-zero value. Upon fitting a model and finding singular β values, it will refit the model the specified number of times, using the initial values from the previous fitting but setting the initial value of singular βs to 0. However, in this case, re-running the analysis produces the same result. A quick check of the capture histories for the first release cohort shows that there was not a single encounter of the first cohort on the last occasion:

```
> dipper$ch[substr(dipper$ch,1,1)==1]
 [1] "1000000" "1000000" "1000000" "1000000" "1000000" "1000000" "1000000"
 [8] "1000000" "1000000" "1010000" "1010000" "1100000" "1100000" "1100000"
[15] "1100000" "1100000" "1100000" "1101110" "1111000" "1111000" "1111100"
[22] "1111110"
```

Thus, with an assumed constant capture probability the best explanation of not seeing any on the seventh occasion is that survival from age 5 to 6 was 0. The parameter is identifiable and is being estimated correctly but it is at a boundary and is not being counted correctly by **MARK**. Moral of the story is to be careful counting parameters (this point has been made at several points in this book). The philosophy incorporated into **RMark** is that it is safer to over-count parameters rather than risk fitting an overly-complex model to sparse data.

The ability of **MARK** to count parameters can be improved by using the sin link which is now supported by **RMark** as long as the resulting design matrix for the parameter is an identity matrix. The sin link can be used for some parameters and a different link for others, so the entire design matrix need not be an identity matrix. If the model formula contains any design or individual covariates then the sin link is not allowed. Also, to use the sin link the formula cannot be additive or use an intercept. If either of the above occurs, an error message will be generated if you specify the sin link. To specify an identity design matrix there must be a 1:1 relationship between the β's and the real parameters. Because **RMark** simplifies the PIMS this can occur even when the group structure is quite complex. As an example, we will use **example.data** which has sex, age and region factors for grouping. Even though there are many parameters in the all-different formulation we can use the sin link with the intercept model (as shown below) because there is one β and one real.

```
> data(example.data)
> example.processed=process.data(example.data,groups=c("sex","age","region"),initial.ages=c(0,1,2))
> example.ddl=make.design.data(example.processed)
> mark(example.processed,example.ddl,model.parameters=list(Phi=list(formula=~1,link="sin")),output=F)
```

For the ~**time** model there is also one β for each real parameter but if we specify the model with the sin link, we get an error message:

```
> mark(example.processed,example.ddl,
      model.parameters=list(Phi=list(formula=~time,link="sin")),output=F)

 Error in make.mark.model(data.proc, title = title, covariates = covariates,  :

 Cannot use sin link with non-identity design matrix
```

The error occurs because ~**time** creates a design matrix with an intercept and a β for each time beyond the first time, so it is *additive* which is not allowed. However, we can specify a design matrix without an intercept using ~**-1 + time** as shown below, or ~**-1+sex**:

```
> mark(example.processed,example.ddl,model.parameters=list(Phi=list(formula=~-1+time,link="sin")),output=F)
> mark(example.processed,example.ddl,model.parameters=list(Phi=list(formula=~-1+sex,link="sin")),output=F)
```

Likewise, we can use the sin link with full interaction models as long as the intercept is removed.

```
> mark(example.processed,example.ddl,model.parameters=
   list(Phi=list(formula=~-1+region:time,link="sin")),output=F)
```

But again you cannot have any additive terms even in the case of adding 2 two-way interactions:

```
> mark(example.processed,example.ddl,model.parameters=
      list(Phi=list(formula=~-1+region:time+sex:time,link="sin")),output=F)

 Error in make.mark.model(data.proc, title = title, covariates = covariates,  :

 Cannot use sin link with non-identity design matrix
```

But the 3-way interaction model can use the sin link:

```
 > mark(example.processed,example.ddl,model.parameters=
       list(Phi=list(formula=~-1+sex:region:time,link="sin")),output=F)
```

C.11. Fixing real parameter values

Parameter confounding presented a situation in which it was useful to fix specific real parameter values and in C.10 we showed how that could be done with the **fixed** argument of the parameter specification list. However, there are other instances in which real parameter values need to be fixed and there are several ways in which fixed parameters can be specified with **RMark**. In addition to parameter confounding, real parameters are typically fixed in circumstances in which there are no data to estimate the parameter (i.e., a structural zero). For example, imagine a scenario where you conducted a "CJS" study in which new animals were only released every other year. In those years with no releases there cannot be any recaptures from that cohort to estimate the parameters. For the limiting case in which only one cohort is released and then followed through time, see discussion about **pim.type** at the end of this section. Another example would be a Multistrata model in which the strata are defined such that some transitions are not possible, so they would be fixed to 0.

There are 2 general approaches to specification of fixed parameters. The first approach was introduced in C.10 using the **fixed** argument which identifies specific parameters by their indices and

specifies their fixed values. The second approach is to delete the rows from the design data for the parameters that are to be fixed at a single default value for each type of parameter (e.g., φ or p). If you need to fix parameters of the same type to different values (e.g., some $p = 0$ and others $p = 1$), you need to use the first approach. The second approach is most useful when all the parameters are being fixed to the same value because of missing data (i.e., structural zero). We will use the dipper data to illustrate how to fix real parameters using these different approaches.

There are 4 different forms for the fixed argument. The first sets all of the parameters of a particular type to the same value. For example, the following poor and non-realistic model would set all of the φ values to 1 for the dipper data.

```
> dipper.processed=process.data(dipper,groups=("sex"),begin.time=1980)
> dipper.ddl=make.design.data(dipper.processed)
> dipper.ddl$p
> Phidot=list(formula=~1)
> Phi.1=list(formula=~1,fixed=1)
> mark(dipper.processed,dipper.ddl,model.parameters=list(Phi=Phi.1))

  Output summary for CJS model
  Name : Phi(~1)p(~1)

  Npar :  1
  -2lnL:  981.2354
  AICc :  983.2449

  Beta
                   estimate        se       lcl        ucl
  Phi:(Intercept)  0.000000 0.0000000  0.000000  0.0000000
  p:(Intercept)   -1.018446 0.0777791 -1.170893 -0.8659991


  Real Parameter Phi
  Group:sexFemale
       1980 1981 1982 1983 1984 1985
  1980    1    1    1    1    1    1
  1981         1    1    1    1    1
  1982              1    1    1    1
  1983                   1    1    1
  1984                        1    1
  1985                             1

  Group:sexMale
       1980 1981 1982 1983 1984 1985
  1980    1    1    1    1    1    1
  1981         1    1    1    1    1
  1982              1    1    1    1
  1983                   1    1    1
  1984                        1    1
  1985                             1
```

Fixing all of the parameters to one value is most useful to simplify the model structure. For example, setting the fidelity parameter (F) in the Burnham model for the case in which the recovery and recapture areas are the same or setting the resight probability (**R** and **RPrime**) in the Barker model to zero to get the Burnham model.

The other forms of the fixed argument involve specifying a set of times, cohorts, ages, groups or generic indices and a set of one or more values. The first 4 are simply short-cuts for the most general approach of specifying indices. Let's start with a generalization of the approach given in C.10 in which we want to fix $p = 0$ in 1982 and 1984 (presumably because of no sampling):

```
> p.time.fixed=list(formula=~time,fixed=list(time=c(1982,1984),value=0))
> mark(dipper.processed,dipper.ddl,model.parameters=list(p=p.time.fixed))

  Real Parameter p
  Group:sexFemale
            1981 1982      1983 1984      1985      1986
  1980 0.9343357    0 0.5387934    0 0.8816249 0.9999979
  1981              0 0.5387934    0 0.8816249 0.9999979
  1982                0.5387934    0 0.8816249 0.9999979
  1983                             0 0.8816249 0.9999979
  1984                               0.8816249 0.9999979
  1985                                         0.9999979

  Group:sexMale
            1981 1982      1983 1984      1985      1986
  1980 0.9343357    0 0.5387934    0 0.8816249 0.9999979
  1981              0 0.5387934    0 0.8816249 0.9999979
  1982                0.5387934    0 0.8816249 0.9999979
  1983                             0 0.8816249 0.9999979
  1984                               0.8816249 0.9999979
  1985                                         0.9999979
```

The same approach will work if you specify certain age, cohort or group values. The use of group is restricted to the group numbers and not the factor variables defining the groups.

Now you would think that the following would work to constrain p for 1982 to 0 and p for 1986 to 1, but it does not (although the programming could be changed) because it expects to have as many values as there are parameters associated with times 1982 and 1986.

```
> p.time.fixed=list(formula=~time,fixed=list(time=c(1982,1986),value=c(0,1)))
> mark(dipper.processed,dipper.ddl,model.parameters=list(p=p.time.fixed))

Lengths of indices and values do not match for fixed parameters for p
Error in make.mark.model(data.proc, title = title, covariates = covariates,
```

That brings us to the final approach which is to specify the parameter indices and the values for those parameters. The indices are the row numbers of the design data for the parameter. For example, in the first 10 rows of the p design data, the indices for 1982 are 2 and 7:

```
  dipper.ddl$p[1:10,]
      group cohort age time Cohort Age Time    sex
  1  Female   1980   1 1981      0   1    0 Female
  2  Female   1980   2 1982      0   2    1 Female
  3  Female   1980   3 1983      0   3    2 Female
  4  Female   1980   4 1984      0   4    3 Female
```

```
 5  Female   1980   5 1985      0   5    4 Female
 6  Female   1980   6 1986      0   6    5 Female
 7  Female   1981   1 1982      1   1    1 Female
 8  Female   1981   2 1983      1   2    2 Female
 9  Female   1981   3 1984      1   3    3 Female
 10 Female   1981   4 1985      1   4    4 Female
```

Now you certainly don't want to look them up and type them in because you will almost certainly make a mistake and it would disable automatic updating of the model if the group structure changed or another occasion was added. The solution is to use a little **R** code to define the set of indices as follows:

```
> p1982.indices=as.numeric(row.names(dipper.ddl$p[dipper.ddl$p$time==1982,]))
> p1982.indices
  [1]  2  7 23 28
> p1986.indices=as.numeric(row.names(dipper.ddl$p[dipper.ddl$p$time==1986,]))
> p1986.indices
  [1]  6 11 15 18 20 21 27 32 36 39 41 42
```

The above code selects the indices which have a time of 1982 and stores it into **p1982.indices** and likewise for 1986. That code will work even if the group or data structure changes as long as the **begin.time** doesn't change but even that part of the code could be automated. Now you don't want to count how many values need to be set to 0 and 1, so again we use some **R** code to do the work:

```
>  p1982.values=rep(0,length(p1982.indices))
>  p1986.values=rep(1,length(p1986.indices))
>  p1982.values
 [1] 0 0 0 0
> p1986.values
  [1] 1 1 1 1 1 1 1 1 1 1 1 1
```

Finally, you can put it all together as follows:

```
> p.time.fixed=list(formula=~time,fixed=list(index=c(p1982.indices,p1986.indices),
                                    value=c(p1982.values,p1986.values)))
> mark(dipper.processed,dipper.ddl,model.parameters=list(p=p.time.fixed))

  Real Parameter p
  Group:sexFemale
            1981 1982      1983      1984     1985 1986
  1980 0.9720207    0 0.8387273 0.8880947 0.938504    1
  1981              0 0.8387273 0.8880947 0.938504    1
  1982                0.8387273 0.8880947 0.938504    1
  1983                          0.8880947 0.938504    1
  1984                                    0.938504    1
  1985                                                1

  Group:sexMale
            1981 1982      1983      1984     1985 1986
  1980 0.9720207    0 0.8387273 0.8880947 0.938504    1
  1981              0 0.8387273 0.8880947 0.938504    1
  1982                0.8387273 0.8880947 0.938504    1
```

```
1983                          0.8880947 0.938504    1
1984                                    0.938504    1
1985                                                1
```

It may help to examine the value for **fixed**, which we can see is a list with the 2 sets of indices (**$index**) pasted (concatenated) together and a set of values (**$value**), which contain 4 zeros for the 1982 parameters and 12 ones for the 1986 parameters.

```
> list(index=c(p1982.indices,p1986.indices),value=c(p1982.values,p1986.values))

 $index
  [1]  2  7 23 28  6 11 15 18 20 21 27 32 36 39 41 42

 $value
  [1] 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1
```

The above approach is completely general and you can use the same pattern and simply change the subset of parameters that are selected. Without showing the results, the following snippet of code could be used to set p in 1982 for females to 0 but for males it sets p in 1984 to 0:

```
> p1982.f=as.numeric(row.names(dipper.ddl$p[dipper.ddl$p$time==1982&
                                            dipper.ddl$p$sex=="Female",]))
> p1984.m=as.numeric(row.names(dipper.ddl$p[dipper.ddl$p$time==1984&
                                            dipper.ddl$p$sex=="Male",]))
> p.time.fixed=list(formula=~time,fixed=list(index=c(p1982.f,p1984.m),value=0))
> mark(dipper.processed,dipper.ddl,model.parameters=list(p=p.time.fixed))
```

Now if the parameters need to be fixed to model structural zeros in the data, then deleting the design data for the parameters representing the missing data is typically the easiest approach. To demonstrate with an example, below we stripped the 1982 cohort from the dipper data and saved it in **xdipper**. After processing the data and making the design data, we deleted the φ and p design data for 1982 by copying all design data other than the data for the 1982 cohort. When the model is run, the default summary shows blanks for parameters with deleted design data. When the model is summarized with the argument **show.fixed=TRUE**, then the default parameter values of $p = 0$ and $\varphi = 1$ are shown for the 1982 cohort.

```
> xdipper=dipper[!substr(dipper$ch,1,3)=="001",]
> xdipper.processed=process.data(xdipper,groups=("sex"),begin.time=1980)
> xdipper.ddl=make.design.data(xdipper.processed)
> xdipper.ddl$p=xdipper.ddl$p[xdipper.ddl$p$cohort!=1982,]
> xdipper.ddl$Phi=xdipper.ddl$Phi[xdipper.ddl$Phi$cohort!=1982,]
> xdipper.model=mark(xdipper.processed,xdipper.ddl)


> summary(xdipper.model,show.fixed=TRUE)

 Real Parameter Phi
 Group:sexFemale
           1980      1981      1982      1983      1984      1985
```

```
1980 0.5886486 0.5886486 0.5886486 0.5886486 0.5886486 0.5886486
1981           0.5886486 0.5886486 0.5886486 0.5886486 0.5886486
1982                     1.0000000 1.0000000 1.0000000 1.0000000
1983                               0.5886486 0.5886486 0.5886486
1984                                         0.5886486 0.5886486
1985                                                   0.5886486

Group:sexMale
          1980      1981      1982      1983      1984      1985
1980 0.5886486 0.5886486 0.5886486 0.5886486 0.5886486 0.5886486
1981           0.5886486 0.5886486 0.5886486 0.5886486 0.5886486
1982                     1.0000000 1.0000000 1.0000000 1.0000000
1983                               0.5886486 0.5886486 0.5886486
1984                                         0.5886486 0.5886486
1985                                                   0.5886486

Real Parameter p
Group:sexFemale
          1981      1982      1983      1984      1985      1986
1980 0.8919246 0.8919246 0.8919246 0.8919246 0.8919246 0.8919246
1981           0.8919246 0.8919246 0.8919246 0.8919246 0.8919246
1982                     0.0000000 0.0000000 0.0000000 0.0000000
1983                               0.8919246 0.8919246 0.8919246
1984                                         0.8919246 0.8919246
1985                                                   0.8919246

Group:sexMale
          1981      1982      1983      1984      1985      1986
1980 0.8919246 0.8919246 0.8919246 0.8919246 0.8919246 0.8919246
1981           0.8919246 0.8919246 0.8919246 0.8919246 0.8919246
1982                     0.0000000 0.0000000 0.0000000 0.0000000
1983                               0.8919246 0.8919246 0.8919246
1984                                         0.8919246 0.8919246
1985                                                   0.8919246
```

Because structural zeros can be a common occurrence with missing cohorts, a function argument **`remove.unused`** was added to **`make.design.data`**. If it is set to **`TRUE`**, then the design data is automatically deleted for any cohorts without any releases. Thus, the example above can be run with the following commands:

```
> xdipper.ddl=make.design.data(xdipper.processed,remove.unused=TRUE)
> xdipper.model=mark(xdipper.processed,xdipper.ddl)
> summary(xdipper.model,show.fixed=TRUE)
```

Some of the parameters have a natural default value (Table C.1) that is assigned if the design data are deleted but on occasion you may want to change the default value or assign a default value to a parameter that has no assigned default value. That is accomplished by setting the argument default in the parameter specification list. In the following rather silly example, the defaults for the above analysis are set to $p = 0.9$ and $\varphi = 0.5$:

```
> xdipper.model=mark(xdipper.processed,xdipper.ddl,
                     model.parameters=list(p=list(default=.9),Phi=list(default=.5)))
> summary(xdipper.model,show.fixed=TRUE)
```

In some cases, you can handle the structural zeros by using a PIM type other than the default "all different". It can be useful to use **pim.type="time"** or **pim.type="constant"** in the call to **make.design.data**. If you choose one of these simpler PIM structures, you cannot use formula for that parameter that is more complex than the structure allows. A constant PIM can be useful to simplify a model by fixing a real parameter at a value ($F = 1$ for Burnham model) or only allowing models with a single parameter to be estimated. A time PIM can be used in a similar situation for triangular PIMS which can be useful with the CJS model for a single release cohort. Using **pim.type="time"** eliminates the need for deleting the unneeded design data and the summary printout of real parameters is limited to a single line. We demonstrate with the dipper data which is restricted to the data from the first release cohort:

```
> data(dipper)
> dipper=dipper[substr(dipper$ch,1,1)=="1",]
> mark(dipper,design.parameters=list(p=list(pim.type="time"),
       Phi=list(pim.type="time")))

  Real Parameter Phi

            1         2         3         4         5         6
  1 0.6043321 0.6043321 0.6043321 0.6043321 0.6043321 0.6043321


  Real Parameter p

            2         3         4         5         6         7
  1 0.8207724 0.8207724 0.8207724 0.8207724 0.8207724 0.8207724
```

Had we not used **pim.type="time"**, then summary would have shown the entire triangular PIM even if the design data were deleted.

C.12. Data Structure and Import for RMark

So far we have only used the dipper data that accompanies **RMark** and have not discussed data requirements. There are numerous other example datasets to demonstrate other models in **RMark** (e.g., **BlackDuck**, **mallard**, **mstrata** and many others). However, to use **RMark** for your own data you need to understand the requirements for the data structure and how to input data. Data for **RMark** must exist in an **R** dataframe with some formatting and naming conventions. There are numerous ways to create dataframes in **R** but we will describe two functions in **RMark** to help in creating the necessary dataframe.

The format for data input with **RMark** is different than with **MARK**, but a function **convert.inp** was written to convert an **.inp** file used for **MARK** to a dataframe for **RMark**. The conversion is necessary because the data format and structures for **MARK** and **RMark** have a fundamental difference in handling groups, as illustrated below. The formats are similar in that each record (row) contains a capture history which can represent one or more animals as specified by the count (freq) and any number of covariates can be tacked on at the end of the record. However, with **MARK**, group structure

is accommodated by having a count (freq) for each group but the data do not contain any information about what was used to construct the groups. The group structure is only represented by group labels. In comparison with **RMark**, each record only represents animals from a single group and the record can contain columns for factor variables that are used to define the group structure.

First we will start with a simple example to show how easy it is to convert an .inp file and then we'll work into more complicated examples. The **\Examples** subdirectory of **MARK** contains a file **pradel.inp** which can be converted to a dataframe for **RMark** and we can look at the first 5 rows with the following commands:

```
> pradel=convert.inp("C:/Program Files/MARK/Examples/pradel.inp")
 pradel[1:5,]
              ch freq
 1 0000000000001   47
 2 0000000000010   36
 3 0000000000011   12
 4 0000000000100   30
 5 0000000000101    8
```

The first argument value "`C:/Program Files/MARK/Examples/pradel.inp`" is the name of the **MARK** **.inp** file which is shown here with the full path and file specification. Make sure to use a single forward slash or two backslashes to separate sub-directories (note: forward slash? backward slash? The differences reflect the convention used in **R** to accommodate different directory naming conventions between Windows on the one hand, and almost everything else on the other). The extension is assumed to be **.inp** but if it is something different, you can specify the extension with the filename (e.g., "**pradel.txt**").

If there is no group structure or covariates as in the above example then the conversion is quite easy, but it does not get much more complicated. Now let's consider the **MARK** example **Pass3MStrata5.inp** which has 2 covariates weight and length and also uses comments to identify each row uniquely. First we will show what happens if you get it wrong:

```
> p3m5=convert.inp("C:/Program Files/MARK/Examples/Pass3MStrata5.inp")

 Error in convert.inp("C:/Program Files/MARK/Examples/Pass3MStrata5.inp") :
 Number of columns in data file does not match group/covariate specification
```

We forgot to specify the 2 covariates that were in the file, so let's try it again:

```
> p3m5=convert.inp("C:/Program Files/MARK/Examples/Pass3MStrata5.inp",
                   covariates=c("weight","length"))
> p3m5[1:5,]
             ch freq weight length
 1 U00000000000    1   1295    548
 2 00000000000U    1   2653    671
 3 000000D0D000    1   1324    528
 4 0000000000W0    1   1415    570
 5 0000D0000000    1    982    500
```

You can see that it ignored the comments contained on each row of the file. If they are unique and we wanted to use them to label the data, we would change it to:

```
> p3m5=convert.inp("C:/Program Files/MARK/Examples/Pass3MStrata5.inp",
            covariates=c("weight","length"),use.comments=TRUE)
> p3m5[1:5,]
                                    ch freq weight length
  1F1-16C-1054 Upper Green   U00000000000    1   1295    548
  1F1-567-2B3A Upper Green   00000000000U    1   2653    671
  1F1-70D-3E7F Desolation    000000D0D000    1   1324    528
  1F1-770-5307 White         0000000000W0    1   1415    570
  1F1-940-3256 Desolation    0000D0000000    1    982    500
```

Had the comments not provided unique labels then the conversion would have failed and the **use.comments** argument would have to be deleted.

Now let's consider how to convert an **.inp** file that contains a group structure. We will use the **MARK** example file **huggins.inp** which has 2 groups. The example doesn't label those groups but we'll assume that the first group was for females and the second for males. To convert the file, we need to specify a value for the argument **group.df** which is a dataframe that specifies the covariates that will be assigned to each group in the **RMark** dataframe. The rows of **group.df** must correspond to the columns of the frequency field in the **.inp** file. In this simple example, there are 2 columns so there will be 2 rows in our dataframe which will be specified as **group.df=data.frame(sex=c("Female","Male"))**.

Let's dissect that command. First **c("Female","Male")** creates a vector by pasting (concatenating) the strings **"Female"** and **"Male"**. Then the vector is assigned to "**sex**" which will be the name in the **R** dataframe for that group factor value. So the commands to convert and look at the first and last few records are:

```
> huggins=convert.inp("C:/Program Files/MARK/Examples/huggins.inp",
                      group.df=data.frame(sex=c("Female","Male")))
> head(huggins)
          ch freq    sex
 1:1 0101010    1 Female
 1:2 0011000    1 Female
 1:3 1001100    1 Female
 1:4 1100101    1 Female
 1:5 0101010    1 Female
 1:6 1011011    1 Female

> tail(huggins)
            ch freq  sex
 2:389 0001111    1 Male
 2:390 0010000    1 Male
 2:391 1000101    1 Male
 2:392 0100000    1 Male
 2:393 1010100    1 Male
 2:394 1001010    1 Male
```

Now let's move onto a more complicated group structure. Below is a table showing the first 4 records from the swift dataset (**multi_group.inp** file) that is described in chapter 6. Groups represent 2 levels of sex (female/male) and 2 levels (good/poor) of colony and because there are 4 groups, there are an equivalent number of frequency fields for each capture history. For example there are 145 Females in

'poor' colonies with the capture history **0010** and 171 Males in 'good' colonies with the same capture history. Here are the first 4 records in the file:

history	*female, good*	*female, poor*	*male, good*	*male, poor*
0010	150	145	171	167;
0011	200	205	179	183;
0100	213	198	131	77;
0101	14	26	32	50;

As shown below, the same 4 records expand to 16 records in the **RMark** format because each record corresponds to a single animal or group of animals. If one or more of the frequencies is zero the record is not needed. While the **MARK** format can be more compact it is less flexible in modifying the group structure. The **RMark** data format can be created from the **MARK** format with the function **convert.inp**. The function call for this example would be:

```
multigroup=convert.inp("multi_group",
   group.df=data.frame(sex=c(rep("Female",2),rep("Male",2)),
   Colony=rep(c("Good","Poor"),2)))
```

Here is the format of the **RMark** dataframe for same **multi_group.inp** data file.

history	*frequency*	**sex**	**colony**
0010	150	female	good
0011	200	female	good
0100	213	female	good
0101	14	female	good
0010	145	female	poor
0011	205	female	poor
0100	198	female	poor
0101	26	female	poor
0010	171	male	good
0011	179	male	good
0100	131	male	good
0101	32	male	good
0010	167	male	poor
0011	183	male	poor
0100	77	male	poor
0101	50	male	poor

The function argument **group.df** specifies the factor variables that will be used to define the 4 groups in the **.inp** file. It is a dataframe and it must contain a record for each group in the left to right order they are given in the **.inp** file. In this file the first 2 groups are for females and the last 2 are for males and the colony type alternates (good, poor, good, poor). Let's dissect the value assigned to **group.df**:

```
data.frame(sex=c(rep("Female",2),rep("Male",2)),colony=rep(c("Good","Poor"),2))
```

The call to function **data.frame** creates an **R** dataframe with 2 columns named **sex** and **colony**. The names can be any valid name and the order in the dataframe is not relevant. Had there been more fields

they could have been added by assigning more columns in the dataframe. What *does* matter is that the columns are all of the same length and for this particular dataframe the order of the rows must match the order of the columns in the **MARK** **.inp** file.

```
sex=c(rep("Female",2),rep("Male ",2))
```

creates a vector of length 4 that is "Female", "Female", "Male", "Male". The function **rep()** creates a vector by *repeating* the first argument value ("Female" or "Male") the number of times specified as the second argument value (2 in this case). The function **c()** concatenates (pastes) together its arguments in the order they are specified. So it sticks together the 2 vectors each with 2 elements into a single vector of length 4.

The code

```
colony=rep(c("Good","Poor"),2)
```

repeats the vector **c("Good","Poor")** twice to yield a vector with 4 elements "Good","Poor","Good", "Poor" which is the order we want.

The resulting **group.df** looks as follows:

```
   sex      Colony
1 Female   Good
2 Female   Poor
3  Male    Good
4  Male    Poor
```

Notice that the values are not shown in quote marks as they were specified. When columns in a dataframe are specified with character strings they are coerced into factor variables by default and that is what we want in this case. What is stored in the dataframe is actually an index to a factor level (numerically 1 or 2 in this case) and what is shown is the label for the factor. This is apparent if we ask for a summary of the fields (columns) or look at just a single column:

```
> summary(group.df)
     sex     Colony
 Female:2    Good:2
 Male  :2    Poor:2

> group.df$sex
[1] Female Female Male   Male Levels: Female Male.
```

By default, levels of a factor variable are assigned in alphabetical order.

These variables are then assigned to the matching capture history and frequency to create a record in the resulting **RMark** dataframe which has been named **multigroup** in this example. The resulting dataframe **multigroup** would look like the representation of the table above if it were sorted by **sex** and **colony** for the first four records.

If the **MARK** **.inp** file also contains individual covariates, the names of these are specified in the covariates argument of **convert.inp**. For example, the call to **convert.inp** for the example file **indcov1** from chapter 11 is:

```
    indcov1=convert.inp("indcov1",covariates=c("cov1","cov2"))
```

The call specifies that the two covariates should be named **cov1** and **cov2** in the **RMark** dataframe. In this example, there was only a single group, so **group.df** was not specified.

The final argument for **convert.inp** is **use.comments** which can be either **TRUE** or **FALSE** (default). Comments in **MARK .inp** files are given as values between /* and */ and these are ignored in the conversion if they span several lines or the whole line is a comment. However, in some cases the comment is used to specify a label for each capture history (e.g., a tag number for the animal) and it may be useful to retain its value in the **RMark** dataframe. If the values of the comments are unique, you can specify **use.comments=TRUE** and it will use the value for the row name of the dataframe. This is shown using the **blckduck.inp** file that accompanies **MARK**:

```
> bd=convert.inp("blckduck",covariates=c("age","weight","winglen","ci"),
    use.comments=TRUE)
```

The first 5 records of the resulting dataframe show the row names as $01, 04, \cdots, 07$ as follows:

```
  > bd[1:5,]
                      ch freq age weight winglen   ci
  01  1100000000000000    1   1   1.16    27.7 4.19
  04  1011000000000000    1   0   1.16    26.4 4.39
  05  1011000000000000    1   1   1.08    26.7 4.04
  06  1010000000000000    1   0   1.12    26.2 4.27
  07  1010000000000000    1   1   1.14    27.7 4.11
```

If you do not have an existing **MARK .inp** file, an **RMark** dataframe can be constructed from a space or tab-delimited text file using the function **import.chdata**. The function specification is:

```
> import.chdata(filename, header = TRUE, field.names = NULL, field.types = NULL)
```

The argument **filename** specifies the path (if not in directory with workspace) and name (with extension) of a text file. The first field in the file should be the capture history string. It cannot contain any spaces or other delimiting characters and it must be named "**ch**". All other fields can be in any order. The group frequency is not needed if each record specifies the data for a single animal. If frequency is in the file, it should be named "**freq**". The first record in the file can be the names of the fields. If the fields are named, the first field must be named **ch** (for capture history) and if frequency is specified it must be named freq. All other field names for individual covariates can be given any valid name. If the first line contains the field names then the argument header should be specified as the default of **TRUE**. If the first line in the file does not contain the field names, then they should be specified as a vector of strings with the argument field.names. Fields other than "**ch**" should be given a specified field type unless all fields should be treated as factor variables. The possible field types are "**f**","**n**" and "**s**" for factor, numeric and the last specifies that the field should be skipped and not imported.

The following are function calls to import the text files for 3 of the examples accompanying **RMark**. The files can be found in **Rdata.zip** in **C:\Program Files\R\R-v.vv\library\RMark\data** where *v.vv* is the **R** version number.

```
> example.data<-import.chdata("example.data.txt",field.types=c("n","f","f","f"))}
```

```
> edwards.eberhardt<-import.chdata("EdwardsandEberhardt.txt",field.names="ch",
                            header=FALSE)

> dipper<-import.chdata("dipper.txt",field.names=c("ch","sex"),header=FALSE)
```

The first imports **example.data** with 4 fields beyond the capture history. The first field is numeric (**weight**) and the last 3 are factor variables (**age**,**sex**,**region**) that are used as grouping variables. The first 6 lines of the file are as follows:

```
ch weight age sex region
1011101  8.3095857 1 M 1
1110000 11.1449917 1 M 2
1000000  4.0252345 1 M 3
1000000  8.6503865 1 M 4
1010000  9.4225103 1 M 1
```

The field names are on the first line of the data file so they are not specified with the **field.names** argument. The next function call is for the Edwards-Eberhardt rabbit data and it has a single field (the capture history) which is not named on the first line, so it is specified with **field.names="ch"** and **header=FALSE**. The last example imports the familiar dipper data which has 2 fields (**capture history** and **sex**) which are specified with the **field.names=c("ch","sex")** and since **sex** is a factor variable, the **field.types** argument need not be specified, but **header=FALSE** is included because the first line does not include field labels.

The above data format and input functions work for most of the models supported by **RMark** with one exception. They do not work with the nest survival model which does not have an encounter history field (**ch**) and requires a much different data format. See the mallard and killdeer datasets for examples of data entry for the nest survival model. For models with an encounter history, the format for **ch** depends on the type of model. For a description of the relevant format see the example(s) provided for each model supported by **RMark** or the model structure description in **MARK**. When the data are processed with the **process.data** function, it checks to make sure that **ch** only contains values that are valid for the chosen analysis model. For example, for **CJS** models, each digit in **ch** can be either a "**0**", "**1**" or "." where "." implies a missing value. In contrast, **ch** for Multistrata models can contain either "**0**" or an alphabetic character which represents the stratum in which it was observed.

C.13. A more organized approach

So far we have introduced **RMark** by typing various commands into **R** and storing the model results in the workspace and then aggregating them into a list. This is a reasonable way to introduce the material but it is not the way we recommend that you conduct your analyses. In this section we will suggest an alternative approach that uses scripts with functions. We recommend this approach for the following reasons:

1. the **R** statements can be stored in a separate text script that can be easily documented
2. the analysis can be easily repeated with the script
3. functions provide an easy way to create a set of models quickly and avoid accidentally aggregating models from different data sets or model types. We will use the swift data set (**aa.inp**) from Chapter 4 as the example.

begin sidebar

Using R functions to ease the workload

R statements can be typed into a text script file and "sourced" into **R** with the "**File | Source R code**" menu selection. **R** expects the script file to have an extension of .R so it is easiest to use it. You can enter and edit the script file with any text editor but you may find it more convenient to work with an editor like TINN-R (**https://sourceforge.net/projects/tinn-r**) which was designed to work with **R**. Such an editor has several advantages including built-in help with **R**, syntax checking and highlighting which helps identify mismatched braces, brackets and parentheses, and automatic sending of scripts or parts of scripts to **R** for execution.

Scripts can contain any valid **R** statements and function calls. So you could simply enter a sequence of statements like we demonstrated in the previous sections. However, we recommend creating a function to define and run the models and then the script contains the definition of the function and the statements to import data and run the function. We recommend using functions because of some of the limitations of **collect.models** and to take advantage of the function **mark.wrapper** that we have not introduced yet. But first we will give a brief description of functions in **R**.

An **R** function is a set of **R** statements that can accept arguments and can return a single value. The **RMark** package is simply a collection of **R** functions with associated help files. All of the built-in **R** functions are in packages but functions can live outside of packages so you can create functions and store them in scripts or in workspaces. So what is the difference between scripts and functions? A *script* is a collection of **R** statements that can be sourced in and they are interpreted in the context of the workspace (**.Rdata**) as if you were typing them at the keyboard. A *function* is subtly different because when a function is executed it effectively creates its own local workspace (called a *frame* in **R**-speak) which contains variables and objects that are defined within the function. The function can use the values of its arguments, the objects it creates and the function can reference any objects from the frame in which it was defined. We will create a simple function that demonstrates this using the **ls()**. A function is composed of a name, an argument list and the body of the function contained in a set of braces ({}). Below we define a very simple function which we call **myls** which has no arguments (no values listed between the parentheses) and the body of the function is a single statement which calls **ls()**:

```
myls=function() { ls() }
```

To illustrate the concept of frames we will call **ls()** and then **myls()** which calls **ls()** within the function.

```
    [1] "aa"                    "aa.models"             "aa.results"
    [4] "df"                    "dipper"                "dipper.cjs.results"
    [7] "dipper.ddl"            "dipper.mod.avg"        "dipper.mod.avg.adj"
   [10] "dipper.Phi.mod.avg"    "dipper.process"        "model.table"
   [13] "myls"                  "p.dot"                 "p.effort.plus.sex"
   [16] "p.Flood"               "p.time"                "p.Time"
   [19] "p.time.fixed"          "p.Timeplussex"         "Phi.ageclass.plus.sex"
   [22] "Phi.dot"               "Phi.Flood"             "Phi.sex"
   [25] "Phi.sex.plus.age"      "Phi.sexplusage"        "Phi.time"
   [28] "summary.mark"
> myls()
character(0)
```

The call to **ls()** shows the contents of the workspace but the call to **ls()** within **myls()** contains no objects because there have been no objects defined within the function. For further illustration we will give **myls()** an argument, define an object and print out some results to show how objects are referenced within functions.

```
myls=function(x) {
                   cat(paste("p.dot = ",p.dot,"\n"))
                   y=x+1
                   p.dot=y
                   cat("p.dot = ",p.dot,"\n") ls()
                  }

> p.dot
$formula ~1

> myls(1)
  p.dot =  ~1
  p.dot =  2
```

```
  [1] "p.dot" "x"     "y"

> p.dot
$formula ~1
```

The function was designed to show that **p.dot** from the workspace which we defined earlier could be referenced from within the function but once a value was assigned to **p.dot** within the function it became an object with its own definition within the function that was independent and did not change the p.dot in the workspace. The call to **ls()** within the function shows that there are 3 objects within the local function "workspace" named **x** (the value of the calling argument) and **y** and **p.dot** defined in the function. Functions return the value of the object in the last line of the body of the function (e.g., result of **ls()** in this case) or more specifically a **return()** statement can be used and multiple values can be returned via a list(). Functions can modify objects in the workspace with the use of the ≪− assignment operator but it is not a recommended programming practice.

Because **ls()** only operates on objects defined *within* the function, functions like **collect.models()** and **mark.wrapper()** can be limited to that range of objects for selection. The function **collect.models()** is not particularly clever and it is possible for it to unknowingly aggregate analyses from different data sets if they have the same name. It does recognize when it is aggregating models of different type (e.g., **CJS** and **POPAN**) and will issue a warning. It will also issue a warning if the name of the processed data varies between models being collected but if the data are different but use the same object name it does not discriminate. However, if the models are collected within a function it will only collect those defined within the function preventing any unforeseen problems. Also, **mark.wrapper** works well within functions to define the set of models to run and we will demonstrate it here with the analysis of the swift data set from chapter 4.

end sidebar

We will use the swift dataset (Chapter 4) to demonstrate the use of scripts with functions. In the swift example, φ is thought to vary by **colony**, by **time**, or by **colony** and **time** (**colony*time**) because one **colony** has been classified as poor and the other as good. Capture probability p is thought to be either constant or vary by time. All of the pairings are considered for a total of 6 models to evaluate. Such a scheme is exactly how **mark.wrapper** was designed to operate. A set of specifications is given for each parameter and all possible combinations of the specifications of the parameters in the model (e.g., φ and p for CJS) are created for analysis. A function **create.model.list** identifies the model specifications by collecting any object named with a parameter name from the particular model followed by a period and any text description can follow the period. This is why we chose to name parameter specifications like **Phi.time** and **p.dot** as we did. Because it will collect any such objects it is best to use **create.model.list** within a function such that it will only collect those defined within the function.

Below we define the script that we created to analyze the swift data. The script imports the data, creates and runs the models, adjusts for over-dispersion and model averages the parameters. We have used comments identified by text following a # sign to document our analysis. We recommend liberal use of comments to help you understand what you were doing and thinking at the time that you created an analysis.

```
# Swift.R
#
# CJS analysis of the swift data from Chapter 4 of Cooch and White
#
# Import data (aa.inp) and convert it from the MARK inp file format to the \textbf{RMark}
# format using the function convert.inp  It is defined with 2 groups:
# Poor and Good to describe the quality of the colony.  This structure is defined
# with the group.df argument of convert.inp.  It expects that the file aa.inp is
# in the same directory as the current workspace.
#
aa=convert.inp("aa",group.df=data.frame(colony=c("Poor","Good")))
#
# Next create the processed dataframe and the design data.  We'll use a group
# variable for colony so it can be used in the set of models for Phi.  Factor
```

```
# variables (covariates with a small finite set of values) are best handled by using
# them to define groups in the data.
#
  aa.process=process.data(aa,model="CJS",groups="colony")
  aa.ddl=make.design.data(aa.process)
#
# Next create the function that defines and runs the set of models and returns
# a marklist with the results and a model.table.  It does not have any arguments
# but does use the aa.process and aa.ddl objects created above in the workspace
# The function create.model.list is the function that creates a dataframe of the
# names of the parameter specifications for each parameter in that type of model.
# If none are given for any parameter, the default specification will be used for
# that parameter in mark.  We used the adjust=FALSE argument because we know that
# the models time in Phi and p have extra parameters so we will accept the parameter
# counts from MARK and not adjust them.  The first argument of mark.wrapper is the
# model list of parameter specifications.  Remaining arguments that are passed to
# mark must be specified using the argument=value specification because the arguments
# of mark were not repeated in mark.wrapper so they must be passed using the
# argument=value syntax.
#
aa.models=function()
{
  Phi.colony=list(formula=~colony)
  Phi.time=list(formula=~time)
  Phi.colony.time=list(formula=~time*colony)
  p.dot=list(formula=~1)
  p.time=list(formula=~time)
  cml=create.model.list("CJS")
  results=mark.wrapper(cml,data=aa.process,ddl=aa.ddl,adjust=FALSE)
  return(results)
}
#
# Next run the function to create the models and store the results in
# aa.results which is a marklist.
#
aa.results=aa.models()
#
# Adjust for estimated overdispersion of chat=1.127
#
aa.results=adjust.chat(1.127,aa.results)
#
# Compute model averaged parameters
#
aa.model.avg.p=model.average(aa.results,"p",vcv=TRUE)
aa.model.avg.Phi=model.average(aa.results,"Phi",vcv=TRUE)
```

The table of model results is the same as that shown in chapter 4.

```
  > aa.results
                          model npar    QAICc DeltaQAICc       weight QDeviance  chat
  2        Phi(~colony)p(~time)    9 330.2689   0.000000 7.058575e-01  99.08106 1.127
  1           Phi(~colony)p(~1)    3 332.1239   1.855043 2.791898e-01 113.75240 1.127
  5             Phi(~time)p(~1)    8 338.1891   7.920167 1.345472e-02 109.19262 1.127
```

```
  6          Phi(~time)p(~time)   13 342.8825  12.613639 1.287360e-03 102.69603 1.127
  3    Phi(~time * colony)p(~1)   15 346.6140  16.345062 1.992654e-04 101.78298 1.127
  4 Phi(~time * colony)p(~time)   20 352.3334  22.064458 1.141513e-05  95.44214 1.127
```

As well the model averaged capture probabilities shown in Chapter 4 are given below and the results are accurate to 7 places to the right of the decimal. Note that **RMark** only provides the unconditional standard error and the confidence interval based on it and does not provide the percentage due to model uncertainty.

```
> aa.model.avg.p$estimates[1:2,]
  par.index  estimate         se       lcl       ucl fixed group cohort age time Cohort Age Time colony
1        57 0.7502636 0.12064459 0.4698781 0.9732465        Good      1   1    2      0   1    0   Good
2        58 0.7230819 0.09321616 0.5118358 0.8667183        Good      1   2    3      0   2    1   Good
```

begin sidebar

Automating annual survey analyses

The swift example with the models we defined above is not a particularly involved analysis and it does not require much additional work with the standard **MARK** interface because each of the models we used are within the pre-defined set of models. However, even in this case, the **RMark** interface does have an advantage if the data set is routinely updated with an additional year of data. If you add a year of data with the standard **MARK** interface, you have to start from scratch to re-create the **MARK** project and the defined set of models; whereas, with the **RMark** interface it would only require re-running the script after changing the data. In some cases, it may be necessary to modify the script but in most cases even that will not be necessary because the PIM and design data structure are recreated automatically with the new data structure that adds another occasion. **R** has numerous ways of importing data including packages that provide direct access into EXCEL and ACCESS databases. This enables creating a script that requires no user intervention after the data are updated in the appropriate database. The ability to run **R** in batch mode with scripts opens the door to developing an interactive user interface that would run **RMark** with **R** and automate the script development. Such a system is currently being used with an **R** package for distance sampling analysis.

end sidebar

C.14. Defining groups with more than one variable

So far the examples we have shown did not really expand on the pre-defined models in the **MARK** interface except for the use of age and cohort. The pre-defined models in **MARK** include group (**g**) as one of the factors but what happens when groups are composed of two or more factor variables? Consider the **multi_group.inp** example described in Chapter 6 which has 6 sampling occasions and groups defined by **colony** and **sex**. If you include **g** in a model for these data, it will fit 4 parameters for **Poor-Female**, **Good-Female**, **Poor-Male**, and **Good-Male**. Fitting **g** is equivalent to fitting ~**colony*sex** which is the full interaction model for **colony** and **sex**. Within the pre-defined models in **MARK** there is no capacity to fit any of the sub-models: ~**colony**, ~**sex**, and ~**colony+sex** and to fit those models you need to create a design matrix which is described in chapter 6. When you jump to **g*t** models, fitting sub-models becomes even more important. What if capture probability varied by time and colony and survival varied by sex and time? Both of these are sub-models of the **g*t** pre-defined model and require a design matrix.

This is where the formula notation and automatic design matrix development starts to become quite useful. Once the group variables are defined, creating the full interaction model and the sub-models requires no more work than any of the other models that we have developed so far.

Below is a script that provides an analysis with a variety of models:

```
# Multi_group.R
#
# CJS analysis of the multi_group data from Chapter 6 of Cooch and White
#
# Import data (multi_group.inp) and convert it from the MARK inp file format to the RMark
# format using the function convert.inp  It is defined with 4 groups:
# Poor-Female, Good-Female, Poor-Male and Good-Male to describe the q
# quality of the colony and 2 sexes.  This structure is defined
# with the group.df argument of convert.inp which has 4 rows and 2 fields sex and colony
#
multigroup=convert.inp("multi_group",
 group.df=data.frame(sex=c(rep("Female",2),rep("Male",2)),colony=rep(c("Good","Poor"),2)))

#
# Next create the processed dataframe and the design data.  We'll use a group
# variable for colony so it can be used in the set of models for Phi.  Factor
# variables (covariates with a small finite set of values) are best handled by using
# them to define groups in the data.
#
multigroup.process=process.data(multigroup,model="CJS",groups=c("sex","colony"))
multigroup.ddl=make.design.data(multigroup.process)
#
# Next create the function that defines and runs the set of models and returns
# a marklist with the results and a model.table.
#
multigroup.models=function()
{
  Phi.colony=list(formula=~colony)
  Phi.sex=list(formula=~sex)
  Phi.sex.plus.colony=list(formula=~sex+colony)
  Phi.sex.time.plus.colony=list(formula=~sex*time+colony)
  p.time=list(formula=~time)
  p.colony.plus.sex=list(formula=~colony+sex)
  p.colony.time=list(formula=~time*colony)
  cml=create.model.list("CJS")
  results=mark.wrapper(cml,data=multigroup.process,ddl=multigroup.ddl)
  return(results)
}
#
# Next run the function to create the models and store the results in
# multigroup.results which is a marklist.
#
multigroup.results=multigroup.models()
#
# Compute model averaged parameters
#
multigroup.model.avg.p=model.average(multigroup.results,"p")
multigroup.model.avg.Phi=model.average(multigroup.results,"Phi")
```

The results table from the run is:

```
> multigroup.results
```

```
                                        model npar     AICc  DeltaAICc     weight  Deviance
12          Phi(~sex * time + colony)p(~time)   13 15440.75   0.000000 0.95888873 100.41075
11 Phi(~sex * time + colony)p(~time * colony)   17 15447.95   7.198505 0.02622000  99.58174
10  Phi(~sex * time + colony)p(~colony + sex)   12 15449.08   8.330000 0.01489127 110.74725
9                  Phi(~sex + colony)p(~time)    7 15907.94 467.182000 0.00000000 579.62086
8         Phi(~sex + colony)p(~time * colony)   11 15912.68 471.927000 0.00000000 576.34862
6                           Phi(~sex)p(~time)    6 15936.95 496.191000 0.00000000 610.63318
5                 Phi(~sex)p(~time * colony)   10 15938.44 497.686000 0.00000000 604.11265
3                       Phi(~colony)p(~time)    6 15996.61 555.860000 0.00000000 670.30226
2               Phi(~colony)p(~time * colony)   10 16000.36 559.606000 0.00000000 666.03275
7          Phi(~sex + colony)p(~colony + sex)    6 16004.56 563.801000 0.00000000 678.24333
4                  Phi(~sex)p(~colony + sex)    5 16031.57 590.818000 0.00000000 707.26243
1                Phi(~colony)p(~colony + sex)    5 16079.25 638.493000 0.00000000 754.93785
```

There is quite a jump in ΔAIC_c (**DeltaAICc**) from model 10 to model 9. This could be from exclusion of ***time** in φ_i or may be due to lack of convergence. Because model 9 is simpler than models 10-12, the latter is unlikely but we will use this as an opportunity to show how you can easily re-run a model using initial values from another model. The following uses the function **rerun.mark** to re-run model 9 using initial values from model 12 and stores the result back into the **marklist** in position 9:

```
> multigroup.results[[9]]=rerun.mark(multigroup.results[[9]],
   data=multigroup.process,ddl=multigroup.ddl,initial=multigroup.results[[12]])
```

A quick look at the summary output reveals the identical AIC_c values so the model converged to the same values. If the value had changed, we would have had to reconstruct the **model.table** as shown later. When we use **model.average** to obtain model-averaged real parameters we get a warning message that model 11 was dropped because some of the beta variances were negative:

```
> multigroup.model.avg.p=model.average(multigroup.results,"p")
Model  11 dropped from the model averaging because one or more beta variances
   are not positive
> multigroup.model.avg.Phi=model.average(multigroup.results,"Phi")
Model  11 dropped from the model averaging because one or more beta variances
   are not positive
```

Negative variances for the β's are symptomatic of something amiss so those models are dropped by default primarily as a way to draw attention to the issue. Negative variances are set to zero in **MARK** so they show up with an SE= 0.00000 in the output and this behavior is mimicked in **RMark**, In this case, the negative variances occur because one of the parameters is at a boundary. φ for females at time 4 is 1 and this probably occurs because of confounding between φ and p for time 4. **MARK** reported 16 of the 17 parameters were estimable and that beta 5 (female time 4) was singular. We can either re-run this model and set **adjust=FALSE** or we can use the function **adjust.parameter.count** to reset the count to 16 as follows:

```
> multigroup.results[[11]]=adjust.parameter.count(multigroup.results[[11]],16)

   Number of parameters adjusted from  17  to  16
   Adjusted AICc=15445.94
   Unadjusted AICc =  15445.95
```

Once the number of parameters has been adjusted the model of table results must be recalculated:

```
  multigroup.results$model.table=model.table(multigroup.results)
  multigroup.results
                                        model npar     AICc  DeltaAICc     weight  Deviance
12          Phi(~sex * time + colony)p(~time)   13 15440.75   0.000000 0.91731475 100.41075
11 Phi(~sex * time + colony)p(~time * colony)   16 15445.94   5.190998 0.06843961  99.58174
10  Phi(~sex * time + colony)p(~colony + sex)   12 15449.08   8.330000 0.01424564 110.74725
9                  Phi(~sex + colony)p(~time)    7 15907.94 467.182000 0.00000000 579.62086
8          Phi(~sex + colony)p(~time * colony)   11 15912.68 471.927000 0.00000000 576.34862
6                           Phi(~sex)p(~time)    6 15936.95 496.191000 0.00000000 610.63318
5                  Phi(~sex)p(~time * colony)   10 15938.44 497.686000 0.00000000 604.11265
3                        Phi(~colony)p(~time)    6 15996.61 555.860000 0.00000000 670.30226
2               Phi(~colony)p(~time * colony)   10 16000.36 559.606000 0.00000000 666.03275
7           Phi(~sex + colony)p(~colony + sex)    6 16004.56 563.801000 0.00000000 678.24333
4                    Phi(~sex)p(~colony + sex)    5 16031.57 590.818000 0.00000000 707.26243
1                 Phi(~colony)p(~colony + sex)    5 16079.25 638.493000 0.00000000 754.93785
```

The parameters can be model averaged across all models by using **drop=FALSE** as follows:

```
> multigroup.model.avg.p=model.average(multigroup.results,"p",drop=FALSE)
Warning message:

Improper V-C matrix for beta estimates. Some variances non-positive.
 in: get.real(model.list[[i]], parameter, design = model.list[[i]]$design.matrix,

> multigroup.model.avg.Phi=model.average(multigroup.results,"Phi",drop=FALSE)
Warning message:

Improper V-C matrix for beta estimates. Some variances non-positive.
 in: get.real(model.list[[i]], parameter, design = model.list[[i]]$design.matrix,
```

A warning message is given about the negative variances and clearly it does not make sense to consider model averaged estimates of φ for time 4 and p for time 5 but the remaining real parameters are unaffected.

C.15. More complex examples

Now we will consider some more complex examples that require more knowledge about designing formulas in situations where the factors are not fully crossed which means that some interactions do not exist in the data structure. We will use the example from Chapter 7 that uses **age.inp**. These data were derived from a study in which only young were marked and released (CJS design) but the young were then recaptured through time as they aged. With such a design not all ages are represented in all years so these factors are not fully crossed. A fully crossed design would have data for each combination of factors. In year 1 of the experiment there are only young birds that were just banded. In year 2 there are birds that are ages 0 and 1, in year 3 there are birds of ages 0 to 2, etc. The general solution to this type of problem is to create dummy variables (numeric 0/1 coding) and create interactions of effects using the **:** operator which includes the interactions without the main effects. This a very useful tool because it allows you to limit the range of an effect to the subset of parameters that have a value of 1 for the dummy variable. To understand this fully, look at the PIM chart in section 7.1.1 which shows an age by time model in which age is limited to 2 classes of young (age 0) and adult (age 1+). The structure shows time varying probabilities for young with indices 1 to 6 and time varying probabilities for adults

with indices 7 to 11. There are no adults at time 1 so there are only 5 survival probabilities for adults and 6 for young.

The PIM chart in 8.1.1 can be created with a formula by creating a 0/1 dummy variable for each age grouping. For example, let's assume that we created a dummy variable called **young** that is 1 for a **young** animal and 0 for an adult and then another variable called **adult** that is 1 for adult and 0 for young. If you construct a formula with the interaction of **young** and **time** (a factor variable), it will create a parameter for each time for **young** animals (indices 1 to 6) and by default it creates an intercept. To demonstrate this we will convert the input file, process the data and create the design data we need:

```
> #
> # Import data from age.inp file with convert.inp
> #
>   age=convert.inp("age")
> #Process data for CJS model
>   age.process=process.data(age,model="CJS")
> #Make default design data
>   age.ddl=make.design.data(age.process)
> #
> # Add a young/adult age field to the design data for Phi which we have named ya.
> # It uses right=FALSE so that the intervals are 0 (young) and 1 to 7 (adult).
> #
>   age.ddl=add.design.data(age.process,age.ddl,"Phi","age",bins=c(0,1,7),right=FALSE,name="ya")
> #
> # Add a field to the Phi design data that is equivalent except that it is a numeric
> # dummy coding variable with value 1 for young and 0 for adult; field is named young
> #
>   age.ddl$Phi$young=0
>   age.ddl$Phi$young[age.ddl$Phi$age==0]=1
> #
> # Likewise add an adult 0/1 numeric field to the Phi design data
> # which is simply =1-young
>   age.ddl$Phi$adult=1-age.ddl$Phi$young
```

Notice that we were able to create the **adult** field from the **young** field by subtraction. Now, let's show what **model.matrix** does with the formula ~**young:time** to give you a more complete understanding. First we will look at the non-simplified PIMS for φ by wrapping the default **mark** model call within a call to **PIMS**:

```
> PIMS(mark(age,output=F),"Phi",simplified=F)
group = Group 1
   1  2  3  4  5  6
1  1  2  3  4  5  6
2     7  8  9 10 11
3       12 13 14 15
4          16 17 18
5             19 20
6                21
```

That shows there are 21 possible different parameters for φ which will match the number of rows in the following design matrix created by **model.matrix**:

```
> model.matrix(~young:time,age.ddl$Phi)
   (Intercept) young:time1 young:time2 young:time3 young:time4 young:time5 young:time6
1            1           1           0           0           0           0           0
2            1           0           0           0           0           0           0
3            1           0           0           0           0           0           0
4            1           0           0           0           0           0           0
5            1           0           0           0           0           0           0
6            1           0           0           0           0           0           0
7            1           0           1           0           0           0           0
8            1           0           0           0           0           0           0
9            1           0           0           0           0           0           0
10           1           0           0           0           0           0           0
11           1           0           0           0           0           0           0
12           1           0           0           1           0           0           0
13           1           0           0           0           0           0           0
14           1           0           0           0           0           0           0
15           1           0           0           0           0           0           0
16           1           0           0           0           1           0           0
17           1           0           0           0           0           0           0
18           1           0           0           0           0           0           0
19           1           0           0           0           0           1           0
20           1           0           0           0           0           0           0
21           1           0           0           0           0           0           1
attr(,"assign")
[1] 0 1 1 1 1 1 1
attr(,"contrasts")
attr(,"contrasts")$time
[1] "contr.treatment"
```

The resulting design matrix is rather simple and with the exception of the intercept it is an identity matrix for indices (rows) 1,7,12,16,19,21 which are the φ parameters for young. The intercept is the φ parameter (β in link space) for adults and the time dependent probabilities for the young are the intercept plus the appropriate column for each time. So let's do the same with the **adult** field:

```
> model.matrix(~adult:time,age.ddl$Phi)
   (Intercept) adult:time1 adult:time2 adult:time3 adult:time4 adult:time5 adult:time6
1            1           0           0           0           0           0           0
2            1           0           1           0           0           0           0
3            1           0           0           1           0           0           0
4            1           0           0           0           1           0           0
5            1           0           0           0           0           1           0
6            1           0           0           0           0           0           1
7            1           0           0           0           0           0           0
8            1           0           0           1           0           0           0
9            1           0           0           0           1           0           0
10           1           0           0           0           0           1           0
11           1           0           0           0           0           0           1
12           1           0           0           0           0           0           0
13           1           0           0           0           1           0           0
14           1           0           0           0           0           1           0
```

```
15          1          0          0          0          0          0          1
16          1          0          0          0          0          0          0
17          1          0          0          0          0          1          0
18          1          0          0          0          0          0          1
19          1          0          0          0          0          0          0
20          1          0          0          0          0          0          1
21          1          0          0          0          0          0          0
attr(,"assign")
[1] 0 1 1 1 1 1 1
attr(,"contrasts")
attr(,"contrasts")$time
[1] "contr.treatment"
```

Notice that the second column in the matrix is all zeros because there are no design data for an adult at time 1. The code in **RMark** simply removes any column containing all zeroes because it is not needed. For the matrix above, that would create 6 parameters for a model that had one constant young survival and 5 time dependent probabilities for adults; whereas ~**young:time** had 7 parameters with a constant adult survival and 6 time dependent probabilities for young.

So what happens if we use ~**young:time + adult:time** to try and construct the equivalent to the PIM chart in 8.1.1? To save space we won't show the entire design matrix but will show the following summaries:

```
> dim(model.matrix(~young:time + adult:time,age.ddl$Phi))
[1] 21 13
> colSums(model.matrix(~young:time + adult:time,age.ddl$Phi))
(Intercept) young:time1 young:time2 young:time3 young:time4 young:time5 young:time6
21           1           1           1           1           1           1
time1:adult time2:adult time3:adult time4:adult time5:adult time6:adult
 0           1           2           3           4           5
```

After deleting the one column of all zeroes, the resulting design matrix will still have 12 columns but there are only 11 unique parameters as shown in the 8.1.1 PIM chart. The solution is to remove the intercept which can be done by adding -1 to the formula. Below we show the same summaries using the correct formula:

```
> dim(model.matrix(~-1+ young:time + adult:time,age.ddl$Phi))
[1] 21 12
> colSums(model.matrix(~-1+young:time + adult:time,age.ddl$Phi))
young:time1 young:time2 young:time3 young:time4 young:time5 young:time6 time1:adult
1           1           1           1           1           1           0
time2:adult time3:adult time4:adult time5:adult time6:adult
1           2           3           4           5
```

After deleting the zero-sum column it will have the appropriate 11 parameters for the design matrix. Let's fit this model and examine the simplified PIM structure and design matrix.

```
> Phi.yaxtime=list(formula=~-1+young:time+adult:time)
> p.dot=list(formula=~1)
> age.Phi.yaxtime.p.dot=mark(age.process,age.ddl,model.parameters=list(Phi=Phi.yaxtime,
      p=p.dot),output=FALSE)
```

```
> PIMS(age.Phi.yaxtime.p.dot,"Phi")
group = Group 1
   1  2  3  4  5  6
1  1  2  3  4  5  6
2     7  3  4  5  6
3        8  4  5  6
4           9  5  6
5             10  6
6                11

>  age.Phi.yaxtime.p.dot$design.matrix[,1:11]
    Phi:young:time1 Phi:young:time2 Phi:young:time3 Phi:young:time4 Phi:young:time5 Phi:young:time6
Phi g1 c1 a0 t1 "1"             "0"             "0"             "0"             "0"             "0"
Phi g1 c1 a1 t2 "0"             "0"             "0"             "0"             "0"             "0"
Phi g1 c1 a2 t3 "0"             "0"             "0"             "0"             "0"             "0"
Phi g1 c1 a3 t4 "0"             "0"             "0"             "0"             "0"             "0"
Phi g1 c1 a4 t5 "0"             "0"             "0"             "0"             "0"             "0"
Phi g1 c1 a5 t6 "0"             "0"             "0"             "0"             "0"             "0"
Phi g1 c2 a0 t2 "0"             "1"             "0"             "0"             "0"             "0"
Phi g1 c3 a0 t3 "0"             "0"             "1"             "0"             "0"             "0"
Phi g1 c4 a0 t4 "0"             "0"             "0"             "1"             "0"             "0"
Phi g1 c5 a0 t5 "0"             "0"             "0"             "0"             "1"             "0"
Phi g1 c6 a0 t6 "0"             "0"             "0"             "0"             "0"             "1"
p g1 c1 a1 t2   "0"             "0"             "0"             "0"             "0"             "0"
       Phi:time2:adult Phi:time3:adult Phi:time4:adult Phi:time5:adult Phi:time6:adult p:(Intercept)
Phi g1 c1 a0 t1 "0"             "0"             "0"             "0"             "0"             "0"
Phi g1 c1 a1 t2 "1"             "0"             "0"             "0"             "0"             "0"
Phi g1 c1 a2 t3 "0"             "1"             "0"             "0"             "0"             "0"
Phi g1 c1 a3 t4 "0"             "0"             "1"             "0"             "0"             "0"
Phi g1 c1 a4 t5 "0"             "0"             "0"             "1"             "0"             "0"
Phi g1 c1 a5 t6 "0"             "0"             "0"             "0"             "1"             "0"
Phi g1 c2 a0 t2 "0"             "0"             "0"             "0"             "0"             "0"
Phi g1 c3 a0 t3 "0"             "0"             "0"             "0"             "0"             "0"
Phi g1 c4 a0 t4 "0"             "0"             "0"             "0"             "0"             "0"
Phi g1 c5 a0 t5 "0"             "0"             "0"             "0"             "0"             "0"
Phi g1 c6 a0 t6 "0"             "0"             "0"             "0"             "0"             "0"
p g1 c1 a1 t2   "0"             "0"             "0"             "0"             "0"             "1"
```

The numbering of the PIM is different but the structure is identical to the PIM chart in 8.1.1, Also, by rearranging the rows of the design matrix you could make it into an identity matrix because each row and each column have only a single 1.

Below is the script that we wrote to do the analysis above and fit other models for comparison. It includes models other than those fitted in Chapter 7.

```
#  markyoung_age.R  - script for fitting models for age.inp in which only young are marked
#
#
# Import data from age.inp file with convert.inp
#
  age=convert.inp("age")
#Process data for CJS model
  age.process=process.data(age,model="CJS")
#Make default design data
  age.ddl=make.design.data(age.process)
#
# Add a young/adult age field to the design data for Phi which we have named ya.
# It uses right=FALSE so that the intervals are 0 (young) and 1 to 7 (adult).
#
```

```
  age.ddl=add.design.data(age.process,age.ddl,"Phi","age",bins=c(0,1,7),right=FALSE,name="ya")
#
# Add a field to the Phi design data that is equivalent except that it is a numeric
# dummy coding variable with value 1 for young and 0 for adult; field is named young
#
  age.ddl$Phi$young=0
  age.ddl$Phi$young[age.ddl$Phi$age==0]=1
#
# Likewise add an adult 0/1 numeric field to the Phi design data which is simply =1-young
#
  age.ddl$Phi$adult=1-age.ddl$Phi$young

markyoung_age.models=function()
{
#Create formulas for Phi
# A constant survival model
  Phi.dot=list(formula=~1)
# A fully age depdendent but time invariant survival model
  Phi.age=list(formula=~age)
# A limited age model (young/adult) but time invariant survival model
  Phi.ya=list(formula=~ya)
# Limited age-time interaction survival model; young vary by time but adult
# survival is time invariant.  The intercept is the adult value
  Phi.yxtime.a=list(formula=~young:time)
# Fully age (young/adult) and time varying survival model with the time effect
# interacting with age. We cannot use ya*time because there are no adults for time1
# The -1 removes the intercept which is not needed because the young:time creates a
# parameter for each time for the young animals and the adult:time creates a parameter
# for each time that has adults.  It is equivalent to a PIM coding for the problem
# but still uses a design matrix.
  Phi.yaxtime=list(formula=~-1+young:time+adult:time)
#Create formulas for p
  p.dot=list(formula=~1)
  p.time=list(formula=~time)
#Create model list
  cml=create.model.list("CJS")
#Run and return complete set of models
  return(mark.wrapper(cml,data=age.process,ddl=age.ddl))
}
# Run analysis function and store in marklist
> markyoung_age.results=markyoung_age.models()
```

Below is the model results table:

```
> markyoung_age.results
                                      model npar     AICc  DeltaAICc     weight Deviance
9                     Phi(~young:time)p(~1)    8 5050.502   0.000000 0.57098900 139.7729
7     Phi(~-1 + young:time + adult:time)p(~1)   12 5051.756   1.254100 0.30500249 132.9714
10                   Phi(~young:time)p(~time)   13 5053.875   3.372900 0.10573331 133.0730
8  Phi(~-1 + young:time + adult:time)p(~time)   17 5057.385   6.883649 0.01827521 128.5015
5                              Phi(~ya)p(~1)    3 5169.603 119.101100 0.00000000 268.9136
1                             Phi(~age)p(~1)    7 5170.234 119.731864 0.00000000 261.5153
2                           Phi(~age)p(~time)   12 5174.617 124.114840 0.00000000 255.8321
6                            Phi(~ya)p(~time)    8 5175.432 124.930200 0.00000000 264.7031
```

```
4                            Phi(~1)p(~time)    7 5385.402 334.900600 0.00000000 476.6840
3                               Phi(~1)p(~1)    2 5418.139 367.637300 0.00000000 519.4538

> #get summary of model 8 to see how it denotes parameter counts and AICc
> summary(markyoung_age.results[[8]])
Output summary for CJS model
Name : Phi(~-1 + young:time + adult:time)p(~time)

Npar :  17  (unadjusted=16)
-2lnL:  5023.183
AICc :  5057.385  (unadjusted=5055.3628)

Beta
                  estimate        se        lcl        ucl
Phi:young:time1 -1.2946578 0.1785905 -1.6446952 -0.9446205
Phi:young:time2  0.2484213 0.1388986 -0.0238200  0.5206626
Phi:young:time3  0.1464425 0.1308574 -0.1100380  0.4029230
Phi:young:time4 -0.8908855 0.1337821 -1.1530983 -0.6286726
Phi:young:time5 -0.8497168 0.1327969 -1.1099988 -0.5894348
Phi:young:time6 -0.9103939 0.0000000 -0.9103939 -0.9103939
Phi:time2:adult  0.2003894 0.3472809 -0.4802813  0.8810600
Phi:time3:adult  1.1011872 0.2531575  0.6049984  1.5973760
Phi:time4:adult  1.0734292 0.2089632  0.6638612  1.4829971
Phi:time5:adult  0.8498674 0.1874681  0.4824300  1.2173048
Phi:time6:adult  1.2672109 0.0000000  1.2672109  1.2672109
p:(Intercept)    0.4519732 0.3418917 -0.2181345  1.1220810
p:time3          0.0452004 0.3796693 -0.6989513  0.7893522
p:time4          0.1410901 0.3679322 -0.5800569  0.8622371
p:time5          0.1825269 0.3691351 -0.5409778  0.9060316
p:time6          0.4714351 0.3766168 -0.2667339  1.2096041
p:time7          0.3346337 0.0000000  0.3346337  0.3346337


Real Parameter Phi

          1         2         3         4         5         6
1 0.2150655 0.5499304 0.7504825 0.7452485 0.7005393 0.7802649
2           0.5617879 0.7504825 0.7452485 0.7005393 0.7802649
3                     0.5365453 0.7452485 0.7005393 0.7802649
4                               0.2909271 0.7005393 0.7802649
5                                         0.2994923 0.7802649
6                                                   0.2869192


Real Parameter p

          2         3         4         5         6         7
1 0.6111083 0.6217949 0.6440677 0.6535092 0.7157361 0.6871023
2           0.6217949 0.6440677 0.6535092 0.7157361 0.6871023
3                     0.6440677 0.6535092 0.7157361 0.6871023
4                               0.6535092 0.7157361 0.6871023
5                                         0.7157361 0.6871023
6                                                   0.6871023
```

```
> # show model.table using parameter counts from MARK
> model.table(markyoung_age.results[1:10],adjust=F)
                                        model npar     AICc DeltaAICc     weight Deviance
9                       Phi(~young:time)p(~1)    8 5050.502    0.0000 0.55330256 139.7729
7     Phi(~-1 + young:time + adult:time)p(~1)   12 5051.756    1.2541 0.29555501 132.9714
10                   Phi(~young:time)p(~time)   13 5053.875    3.3729 0.10245821 133.0730
8  Phi(~-1 + young:time + adult:time)p(~time)   16 5055.363    4.8611 0.04868422 128.5015
1                              Phi(~age)p(~1)    6 5168.224  117.7226 0.00000000 261.5153
5                               Phi(~ya)p(~1)    3 5169.603  119.1011 0.00000000 268.9136
2                            Phi(~age)p(~time)   11 5172.601  122.0989 0.00000000 255.8321
6                             Phi(~ya)p(~time)    8 5175.432  124.9302 0.00000000 264.7031
4                              Phi(~1)p(~time)    7 5385.402  334.9006 0.00000000 476.6840
3                                 Phi(~1)p(~1)    2 5418.139  367.6373 0.00000000 519.4538
```

This final results table has the same values as the equivalent table in Chapter 7 except that it contains more models including the best models.

Now let's take the next step presented in chapter 7 and consider the situation in which both young and adults are marked and released. The primary goal of this exercise is to evaluate whether adult survival differs for the 2 groups: marked as young versus marked as adult.

```
age_ya=convert.inp("age_ya",group.df=data.frame(age=c("Young","Adult")))
# Process data for CJS model; an initial age is defined as 1 for adults and 0
# for young.  They are assigned in that order because they are assigned in order of
# the factor variable which is alphabetical with adult before young.  It does not
# matter that adults could be a mixture of ages because we will only model young (0)
# and adult (1+).
age_ya.process=process.data(age_ya,group="age",initial.age=c(1,0))
# Make the default design data
age_ya.ddl=make.design.data(age_ya.process)
#
# Add a young/adult age field to the design data for Phi which we have named ya.
# It uses right=FALSE so that the intervals are 0 (young) and 1 to 7 (adult).
#
age_ya.ddl=add.design.data(age_ya.process,age_ya.ddl,"Phi","age",bins=c(0,1,7),
          right=FALSE,name="ya")
#
# Next create a dummy field called marked.as.adult which is 0 for the group
# marked as young and 1 for the group marked as adults.
#
age_ya.ddl$Phi$marked.as.adult=0
age_ya.ddl$Phi$marked.as.adult[age_ya.ddl$Phi$group=="Adult"]=1
```

Look through the design data for φ so you understand how each of the added fields are defined. Pay particular attention to the difference between the **ya** field and **marked.as.adult**. The field **ya** represents age classes and they change over time for an individual marked and released as young whereas the **marked.as.adult** is a dummy variable for the grouping and it is static.

```
> age_ya.ddl$Phi
   group cohort age time Cohort Age Time initial.age.class    ya marked.as.adult
1  Adult      1   1    1      0   1    0             Adult [1,7]               1
2  Adult      1   2    2      0   2    1             Adult [1,7]               1
3  Adult      1   3    3      0   3    2             Adult [1,7]               1
```

```
4  Adult      1   4    4      0   4    3            Adult [1,7]              1
5  Adult      1   5    5      0   5    4            Adult [1,7]              1
6  Adult      1   6    6      0   6    5            Adult [1,7]              1
7  Adult      2   1    2      1   1    1            Adult [1,7]              1
8  Adult      2   2    3      1   2    2            Adult [1,7]              1
9  Adult      2   3    4      1   3    3            Adult [1,7]              1
10 Adult      2   4    5      1   4    4            Adult [1,7]              1
11 Adult      2   5    6      1   5    5            Adult [1,7]              1
12 Adult      3   1    3      2   1    2            Adult [1,7]              1
13 Adult      3   2    4      2   2    3            Adult [1,7]              1
14 Adult      3   3    5      2   3    4            Adult [1,7]              1
15 Adult      3   4    6      2   4    5            Adult [1,7]              1
16 Adult      4   1    4      3   1    3            Adult [1,7]              1
17 Adult      4   2    5      3   2    4            Adult [1,7]              1
18 Adult      4   3    6      3   3    5            Adult [1,7]              1
19 Adult      5   1    5      4   1    4            Adult [1,7]              1
20 Adult      5   2    6      4   2    5            Adult [1,7]              1
21 Adult      6   1    6      5   1    5            Adult [1,7]              1
22 Young      1   0    1      0   0    0            Young [0,1)              0
23 Young      1   1    2      0   1    1            Young [1,7]              0
24 Young      1   2    3      0   2    2            Young [1,7]              0
25 Young      1   3    4      0   3    3            Young [1,7]              0
26 Young      1   4    5      0   4    4            Young [1,7]              0
27 Young      1   5    6      0   5    5            Young [1,7]              0
28 Young      2   0    2      1   0    1            Young [0,1)              0
29 Young      2   1    3      1   1    2            Young [1,7]              0
30 Young      2   2    4      1   2    3            Young [1,7]              0
31 Young      2   3    5      1   3    4            Young [1,7]              0
32 Young      2   4    6      1   4    5            Young [1,7]              0
33 Young      3   0    3      2   0    2            Young [0,1)              0
34 Young      3   1    4      2   1    3            Young [1,7]              0
35 Young      3   2    5      2   2    4            Young [1,7]              0
36 Young      3   3    6      2   3    5            Young [1,7]              0
37 Young      4   0    4      3   0    3            Young [0,1)              0
38 Young      4   1    5      3   1    4            Young [1,7]              0
39 Young      4   2    6      3   2    5            Young [1,7]              0
40 Young      5   0    5      4   0    4            Young [0,1)              0
41 Young      5   1    6      4   1    5            Young [1,7]              0
42 Young      6   0    6      5   0    5            Young [0,1)              0
>
```

Before we go too far with this example, we'll show the simplified PIMS for the ~**ya*time** model which we could not fit in the previous example but we can fit now because adults were marked at time 1.

```
> PIMS(mark(age_ya.process,age_ya.ddl,model.parameters=list(Phi=list(formula=~ya*time)),
output=F),"Phi")

group = ageAdult
   1  2  3  4  5  6
1  1  2  3  4  5  6
2     2  3  4  5  6
3        3  4  5  6
4           4  5  6
```

```
5              5  6
6                 6
group = ageYoung
   1  2  3  4  5  6
1  7  2  3  4  5  6
2     8  3  4  5  6
3        9  4  5  6
4          10  5  6
5             11  6
6                12
```

The numbering is slightly different than what is shown in the second and final sets of PIMS from section 8.1.2, but if you look closely you'll see that the structure is identical with survival varying over time and interacting with age as defined by young/adult age classes. Hmm, that is quite close to what we want for the structure to evaluate whether adult survival is different between the 2 groups. All we really need to do is add **marked.as.adult** to the formula. Let's fit that model for φ and the sub-model given above and assume that capture probability varies by group but is time-invariant:

```
age_ya.models=function() {
Phi.ya.time.plus.marked.as.adult=list(formula=~ya*time+marked.as.adult)
Phi.ya.time=list(formula=~ya*time)
p.marked.as.adult=list(formula=~marked.as.adult)
cml=create.model.list("CJS")
results=mark.wrapper(cml,data=age_ya.process,ddl=age_ya.ddl,output=FALSE)
return(results) }

age_ya.results=age_ya.models()

Variable  marked.as.adult  used in formula is not defined in data
Error in make.mark.model(data.proc, title = title, covariates =
covariates,  :

Variable  marked.as.adult  used in formula is not defined in data
Error in make.mark.model(data.proc, title = title, covariates =
covariates,  :

No mark models found Error in collect.models() :
```

What did we do wrong? We defined **marked.as.adult** and the spelling and punctuation is correct. You will make this mistake which is why we showed it. The error message could be made better because it does not tell you where the problem occurs, but remember that design data is specific to each parameter and we only defined the **marked.as.adult** field for φ but we just used it above for the formula for p. That is the problem. One solution would be to use **~group** for the formula for p because that will give the same model with a slightly different parameterization. Another solution is to create the design data as follows and re-run the analysis:

```
> age_ya.ddl$p$marked.as.adult=0
> age_ya.ddl$p$marked.as.adult[age_ya.ddl$p$group=="Adult"]=1
> age_ya.results=age_ya.models()
```

```
> age_ya.results
                                                    model npar     AICc DeltaAICc    weight Deviance
2 Phi(~ya * time + marked.as.adult)p(~marked.as.adult)   15 13846.48     0.000 0.5304622 274.5737
1                  Phi(~ya * time)p(~marked.as.adult)   14 13846.72     0.244 0.4695378 276.8261
```

Did we get the models and parameter counts correct? With the ~**ya*time** model shown in the final set of PIMS in 8.1.2 there are 12 parameters for φ and for our model with p there are 2 parameters (one for each group) so that is 14 and it matches the count for model 1. Our model 2 adds a single parameter for φ so that makes 15 also matching the results. If we look at the simplified PIMS for φ with model 2 we see that the structure matches the PIMS laid out for this problem with 17 indices, but they are not numbered in the same order:

```
> PIMS(age_ya.results[[2]],"Phi")
group = ageAdult
   1  2  3  4  5  6
1  1  2  3  4  5  6
2     2  3  4  5  6
3        3  4  5  6
4           4  5  6
5              5  6
6                 6
group = ageYoung
   1  2  3  4  5  6
1  7  8  9 10 11 12
2    13  9 10 11 12
3       14 10 11 12
4          15 11 12
5             16 12
6                17
```

The design matrix also does not match the one shown in 8.1.2 because the rows are ordered differently and the effects are parameterized differently so the betas will be different but the real parameters would be the same. The design matrix shown below is a cosmetically edited version of the contents contained in **age_ya.results[[2]]$design.matrix** to make it more visually apparent. The design matrix is stored as a matrix of strings so the "" were removed, the **marked.as.adult** column was moved over, the column headers were renamed to use adult rather than the factor **ya:[1,7]**. The intercept (the first column) is for young- time1 which is apparent when you see that row 7 (the index for this parameter) is the one with a single 1 in the row. The second column is the additive age-effect for adult survival and the third column is the **marked.as.adult** effect which is 1 for only the first 6 rows (indices 1-6). Columns 4-8 are baseline time effects for times 2 to 6. Finally, columns 9-13 are the interaction of time with age for adults. All of these columns would be the same for model 1 except that column 3 would not be included.

```
Adult   Adult   1   1   1   1   0   0   0   0   0   0   0   0   0   0
Adult   Adult   2   1   1   1   1   0   0   0   0   1   0   0   0   0
Adult   Adult   3   1   1   1   0   1   0   0   0   0   1   0   0   0
Adult   Adult   4   1   1   1   0   0   1   0   0   0   0   1   0   0
Adult   Adult   5   1   1   1   0   0   0   1   0   0   0   0   1   0
Adult   Adult   6   1   1   1   0   0   0   0   1   0   0   0   0   1
```

```
Young   Young   7   1   0   0   0   0   0   0   0   0   0   0   0   0
Young   Adult   8   1   1   0   1   0   0   0   0   1   0   0   0   0
Young   Adult   9   1   1   0   0   1   0   0   0   0   1   0   0   0
Young   Adult   10  1   1   0   0   0   1   0   0   0   0   1   0   0
Young   Adult   11  1   1   0   0   0   0   1   0   0   0   0   1   0
Young   Adult   12  1   1   0   0   0   0   0   1   0   0   0   0   1
Young   Young   13  1   0   0   1   0   0   0   0   0   0   0   0   0
Young   Young   14  1   0   0   0   1   0   0   0   0   0   0   0   0
Young   Young   15  1   0   0   0   0   1   0   0   0   0   0   0   0
Young   Young   16  1   0   0   0   0   0   1   0   0   0   0   0   0
Young   Young   17  1   0   0   0   0   0   0   1   0   0   0   0   0
```

It is also useful to distinguish here between TSM (time since marking) and age models. This distinction is made based on the initial age that is assigned to groups. If the initial ages for the groups are identical (and technically 0) then age in the design data is really TSM. Age and TSM are the same when everything is the same age at marking like in the example when only young were marked. If you assign different initial ages to groups to represent actual age, you can still define a TSM field in the design data as age-initial age but make sure to use numeric values like Age or convert factors to numeric values to do the calculation.

Let's go back to the dipper data to show some more complications that can arise when the design is not fully crossed. In this case, we will assume that dippers are all released at age 0 and we expect that survival is time dependent for young (age 0) but not for all adults (1+). Also, we expect age differences in adult survival and we expect that the age differences might be different for males and females. Also, we expect that adult capture probability changes when they reach age 2 for females and age 3 for males. This is most likely bogus for dippers but then again it is just an example. So how do we go about building a set of models? First, we need to set up the design data that we need for the structure that we have identified. The following code processes the data, makes the default design data and then creates fields adult (0/1) and young (0/1) in the ψ design data and the variable shift (0/1) in p which was defined to create a sex-specific timing of a shift in capture probability possibly associated with the onset of breeding age.

```
> dipper.processed=process.data(dipper,begin.time=1980,groups="sex")
> dipper.ddl=make.design.data(dipper.processed)
> dipper.ddl$Phi$adult=0
> dipper.ddl$Phi$adult[dipper.ddl$Phi$age>=1]=1
> dipper.ddl$Phi$adult[dipper.ddl$Phi$Age>=1]=1
> dipper.ddl$Phi$young=1- dipper.ddl$Phi$adult
> dipper.ddl$Phi
> dipper.ddl$p$shift=0
> dipper.ddl$p$shift[dipper.ddl$p$Age>=3&dipper.ddl$p$sex=="Male"]=1
> dipper.ddl$p$shift[dipper.ddl$p$Age>=2&dipper.ddl$p$sex=="Female"]=1
> dipper.ddl$p
```

With these fields defined we can consider how to construct formula for various models that we propose. First we will consider the φ models and we will use the R functions **model.matrix** and **colSums** to examine how the model is constructed. Using **model.matrix** within **colSums** will show the columns in the design matrix and if they are non-zero. For example, if we want time dependent survival for the young we could do as follows:

```
> colSums(model.matrix(~young:time,dipper.ddl$Phi))
(Intercept) young:time1980 young:time1981 young:time1982 young:time1983 young:time1984 young:time1985
         42              2              2              2              2              2              2
```

This formula would create 6 parameters for young survival and then an intercept which would apply to adults which would have a constant survival. If we wanted to add an age and sex dependent survival for adults it would look as follows:

```
> colSums(model.matrix(~young:time+adult:age:sex,dipper.ddl$Phi))
         (Intercept)       young:time1980       young:time1981       young:time1982       young:time1983
                  42                    2                    2                    2                    2
      young:time1984       young:time1985 adult:age0:sexFemale adult:age1:sexFemale adult:age2:sexFemale
                   2                    2                    0                    5                    4
adult:age3:sexFemale adult:age4:sexFemale adult:age5:sexFemale   adult:age0:sexMale   adult:age1:sexMale
                   3                    2                    1                    0                    5
  adult:age2:sexMale   adult:age3:sexMale   adult:age4:sexMale   adult:age5:sexMale
                   4                    3                    2                    1
```

However, it has 17 non-zero columns but we only need 16 parameters (6 for age 0 and 5 each for the ages 1-5 for both male and female). The solution as noted above was to use the -1 to remove the intercept to get 16 parameters:

```
> colSums(model.matrix(~-1+young:time+adult:age:sex,dipper.ddl$Phi))
      young:time1980       young:time1981       young:time1982       young:time1983       young:time1984
                   2                    2                    2                    2                    2
      young:time1985 adult:age0:sexFemale adult:age1:sexFemale adult:age2:sexFemale adult:age3:sexFemale
                   2                    0                    5                    4                    3
adult:age4:sexFemale adult:age5:sexFemale   adult:age0:sexMale   adult:age1:sexMale   adult:age2:sexMale
                   2                    1                    0                    5                    4
  adult:age3:sexMale   adult:age4:sexMale   adult:age5:sexMale
                   3                    2                    1
```

Now, what if we wanted a model with age effects and an additive sex effect solely for adults:

```
> colSums(model.matrix(~-1+young:time+adult:age+adult:sex,dipper.ddl$Phi))
young:time1980 young:time1981 young:time1982 young:time1983 young:time1984 young:time1985     adult:age0
             2              2              2              2              2              2              0
    adult:age1     adult:age2     adult:age3     adult:age4     adult:age5  adult:sexMale
            10              8              6              4              2             15
```

That works as expected with 12 non-zero columns for the 12 parameters (6 for young, 5 for ages and 1 additive sex effect (male) for the adult age classes.

However, if we wanted an additive sex effect for each age including young, things go awry:

```
> colSums(model.matrix(~-1+young:time+adult:age+sex,dipper.ddl$Phi))
     sexFemale        sexMale young:time1980 young:time1981 young:time1982 young:time1983 young:time1984
            21             21              2              2              2              2              2
young:time1985     adult:age0     adult:age1     adult:age2     adult:age3     adult:age4     adult:age5
             2              0             10              8              6              4              2
```

because the -1 does not remove the intercept and it simply changes the design matrix to have separate intercepts for each sex and we end up with 13 parameters instead of 12 as above. Although it will not affect **model.matrix**, the solution for **RMark** is to set the argument **remove.intercept=TRUE** in the

parameter specification as shown below. That will force removal of the intercept and can always be used in place of the -1 in a formula. If you use **remove.intercept=TRUE**, do not use the -1 in the formula.

On the next page is the script for this analysis which examines a sequence of models for φ including those above and a sequence for p including the shift in p. Given that this example was contrived it should be surprising that these imaginary models were not particularly good ones, but we show the results to demonstrate that the number of parameters were correct.

```
do.complicated.dipper.models=function()
{
# retrieve data, process it for CJS model and make default design data
    data(dipper)
    dipper.processed=process.data(dipper,begin.time=1980,groups="sex")
    dipper.ddl=make.design.data(dipper.processed)
# create additional Phi fields adult and young
    dipper.ddl$Phi$adult=0
    dipper.ddl$Phi$adult[dipper.ddl$Phi$Age>=1]=1
    dipper.ddl$Phi$young=1- dipper.ddl$Phi$adult
# create additional p field for sex-specific shift in p at "breeding" age
    dipper.ddl$p$shift=0
    dipper.ddl$p$shift[dipper.ddl$p$Age>=3&dipper.ddl$p$sex=="Male"]=1
    dipper.ddl$p$shift[dipper.ddl$p$Age>=2&dipper.ddl$p$sex=="Female"]=1
# define models for Phi
    Phi.dot=list(formula=~1)
    Phi.ytime=list(formula=~young:time)
    Phi.ytime.plus.adultxagexsex=list(formula=~young:time+adult:age:sex,remove.intercept=TRUE)
    Phi.ytime.plus.adultxage.plussex=list(formula=~young:time+adult:age+sex,remove.intercept=TRUE)
    Phi.ytime.plus.adultxage.plusadultxsex=list(formula=~young:time+adult:age+adult:sex,
                                                remove.intercept=TRUE)
# define models for p
    p.dot=list(formula=~1)
    p.time=list(formula=~time)
    p.shift=list(formula=~shift)
    p.shiftxsex=list(formula=~shift*sex)
# create model list
    cml=create.model.list("CJS")
# run and return models
    return(mark.wrapper(cml,data=dipper.processed,ddl=dipper.ddl))
}

complicated.results=do.complicated.dipper.models()
```

```
complicated.results
```

	model	npar	AICc	DeltaAICc	weight	Deviance
1	Phi(~1)p(~1)	2	670.8660	0.000000	5.877454e-01	84.36055
2	Phi(~1)p(~shift)	3	672.8926	2.026520	2.133713e-01	84.35857
5	Phi(~young:time)p(~1)	8	674.6677	3.801640	8.783621e-02	75.84524
3	Phi(~1)p(~shift * sex)	5	675.9918	5.125757	4.530490e-02	83.37182
6	Phi(~young:time)p(~shift)	9	676.7273	5.861220	3.136472e-02	75.81745
4	Phi(~1)p(~time)	7	678.7481	7.882080	1.141872e-02	82.00306
9	Phi(~young:time + adult:age + adult:sex)p(~1)	13	679.7517	8.885695	6.913294e-03	70.39112
7	Phi(~young:time)p(~shift * sex)	11	680.1781	9.312101	5.585887e-03	75.06334
13	Phi(~young:time + adult:age + sex)p(~1)	13	681.2700	10.403965	3.235913e-03	71.90939

```
10 Phi(~young:time + adult:age + adult:sex)p(~shift)   14 681.7379 10.871858 2.560916e-03 70.23888
8                           Phi(~young:time)p(~time)   13 682.5149 11.648835 1.736508e-03 73.15426
14       Phi(~young:time + adult:age + sex)p(~shift)   14 683.3693 12.503268 1.132763e-03 71.87029
17             Phi(~young:time + adult:age:sex)p(~1)   17 684.4892 13.623220 6.470601e-04 66.51214
11 Phi(~young:time+adult:age+adult:sex)p(~shift*sex)   16 684.9887 14.122623 5.040812e-04 69.18147
18         Phi(~young:time + adult:age:sex)p(~shift)   18 686.5849 15.718830 2.269283e-04 66.42716
15 Phi(~young:time + adult:age + sex)p(~shift * sex)   16 687.0127 16.146713 1.832209e-04 71.20556
12  Phi(~young:time + adult:age + adult:sex)p(~time)   18 687.9284 17.062380 1.159152e-04 67.77071
16        Phi(~young:time + adult:age + sex)p(~time)   18 689.2101 18.344070 6.106960e-05 69.05240
19   Phi(~young:time + adult:age:sex)p(~shift * sex)   20 689.8623 18.996264 4.407607e-05 65.31111
20          Phi(~young:time + adult:age:sex)p(~time)   22 692.6151 21.749106 1.112835e-05 63.62686
```

C.16. Individual covariates

As promised, we will now divulge the fourth trick in **RMark** which was needed to encompass individual covariates. You do not need to know how this trick works to use **RMark** and we are only describing it here in case someone wanted to use it in another similar application. If you look through the help file for `make.mark.model` you will see that there are arguments for a parameter specification called `component` and `component.name`. These arguments were included before the fourth trick was discovered and included. Now they are no longer needed. Those arguments were used to create additional columns that were pasted onto the design matrix to include individual covariates. This was done because individual covariates are entered into the design matrix for **MARK** as a string which contains the name of the covariate rather than 0 or 1 or other numeric value. There is no direct way to use `model.matrix` to do add these covariate names - thus the trick.

When **RMark** encounters an individual covariate (a name not in the design data), it creates a dummy variable in the design data for that covariate. The covariate name is used for the dummy variable name and it is given the value 1 for each row in the design data. Then the entire formula with the individual covariate and the modified design data is passed to model.matrix to create the design matrix which is only partially complete. **RMark** then processes the design matrix further to add the covariate names for **MARK**. Any columns with names that contain any individual covariate are modified in the following way: 1) any 0 values are left as is, 2) any value of 1 is changed to a string with the name of the covariate, and 3) if the value is neither 1 or 0, then it uses the product construct used in **MARK** design matrices and the value is replaced with the string "`product(value,covariate_name)`". The final step enables the use of formula containing interactions of individual covariates and design data covariates.

There is actually one more step that **RMark** does to enable time-varying covariates. If you use an individual covariate name that does not exist in the data, then it will look for variables that have that name as the prefix and a sequence of suffixes that match the values of the time variable in the design data for that particular parameter. This means that the variable names have to be constructed in a fashion that is consistent with the value chosen for begin.time and which is consistent with the labeling of times which is different for interval parameters such as φ and occasion parameters like p. If **RMark** finds a set of covariates that are properly named, then it constructs the design matrix using the covariate names that are appropriate for each row in the design matrix based on the value of the time field for that specific parameter.

Well with that said there is not much more to say about individual covariates except to show some examples that demonstrate how they are used in formula and how covariate-specific real parameter estimates can be computed after the model is fitted. To do that, we will continue to abuse the dipper data and create some imaginary weight data which was the weight of the bird at the time of first capture. We

will fit models in which weight affects survival for all times for both sexes (**weight**) and then with a sex effect and sex-weight interaction (**sex*weight**). We will also show how the affect of the covariate can be limited to the first survival post-capture (**young:weight**). The following is the script that examines these and other models. Comments are given to explain each φ model.

```
# retrieve data, create some imaginary weight data using a random normal
> data(dipper)
> dipper$weight=rnorm(294,10,2)
> do.dipper.covariate.example=function()
{
# process the data for CJS model and make default design data
    dipper.processed=process.data(dipper,begin.time=1980,groups="sex")
    dipper.ddl=make.design.data(dipper.processed)
# create additional Phi fields adult and young
    dipper.ddl$Phi$adult=0
    dipper.ddl$Phi$adult[dipper.ddl$Phi$Age>=1]=1
    dipper.ddl$Phi$young=1- dipper.ddl$Phi$adult
# define models for Phi
    Phi.dot=list(formula=~1)
# weight only for all survivals
    Phi.weight=list(formula=~weight)
# sex and sex-dependent slope for weight
    Phi.weight.x.sex=list(formula=~weight*sex)
# same intercept for male/female with a sex-dependent slope for weight
    Phi.weight.sex=list(formula=~weight:sex)
# effect of weight only for first time post-capture; if you exclude the
# adult term, then an adult would have the intercept survival which would
# be the value for weight=0
    Phi.weight.plus.sex=list(formula=~adult + young:weight)
# define models for p
    p.dot=list(formula=~1)
    p.time=list(formula=~time)
# create model list
    cml=create.model.list("CJS")
# run and return models
    return(mark.wrapper(cml,data=dipper.processed,ddl=dipper.ddl))
    }
> covariate.results=do.dipper.covariate.example()
```

The results really do not matter because the example and data are bogus, but it is useful to examine the resulting design matrix that was constructed for some of these models. You can look at the simplified design matrix easily as follows:

```
> covariate.results[[3]]$design.matrix
                          Phi:(Intercept) Phi:weight p:(Intercept)
Phi gFemale c1980 a0 t1980 "1"             "weight"   "0"
p gFemale c1980 a1 t1981   "0"             "0"        "1"

> covariate.results[[5]]$design.matrix
                          Phi:(Intercept) Phi:adult Phi:young:weight p:(Intercept)
```

```
Phi gFemale c1980 a0 t1980 "1"            "0"       "weight"         "0"
Phi gFemale c1980 a1 t1981 "1"            "1"       "0"              "0"
p gFemale c1980 a1 t1981   "0"            "0"       "0"              "1"

> covariate.results[[7]]$design.matrix
                          Phi:(Intercept) Phi:weight:sexFemale Phi:weight:sexMale p:(Intercept)
Phi gFemale c1980 a0 t1980 "1"            "weight"            "0"                "0"
Phi gMale c1980 a0 t1980   "1"            "0"                 "weight"           "0"
p gFemale c1980 a1 t1981   "0"            "0"                 "0"                "1"

> covariate.results[[9]]$design.matrix
                          Phi:(Intercept) Phi:weight Phi:sexMale Phi:weight:sexMale p:(Intercept)
Phi gFemale c1980 a0 t1980 "1"            "weight"   "0"         "0"                "0"
Phi gMale c1980 a0 t1980   "1"            "weight"   "1"         "weight"           "0"
p gFemale c1980 a1 t1981   "0"            "0"        "0"         "0"                "1"
```

These are the simplified design matrices which are used after the PIMS have been recoded from all-different to the unique values. The non-simplified design matrix would contain 42 rows for φ and 42 rows for p. Notice that the resulting number of rows in the simplified design matrix depends on the formulas used which determine the unique number of parameters required.

It is useful to examine the design matrix to make sure you get the model you think that you specified with the formula. Even though **RMark** creates the PIMS and design matrix for you, it does not mean that you can shut off your brain and stop thinking. As an example, it would be very easy to make a mistake and specify the one model as ~**young:weight**. At first glance that might seem to do what you want to restrict the effect of weight to the first capture φ and it does do that but it also stupidly assigns adult survival to the intercept which is the value where **weight=0**. Even though **RMark** removes the drudgery of creating design matrices, it does not eliminate the possibility of making a mistake by incorrect specification of the model. Examining the design matrix and using **model.matrix** (if there are no individual covariates) is the best way to prevent those mistakes.

Once you have fitted models using individual covariates, you will often want to compute predicted values at one or more covariate values. There are several functions to do this including **compute.real** but the most complete and easiest to use is **covariate.predictions**. Below we compute the value of survival for young in 1980 (**index=1**) for a range of values for weight and then plot the predicted values with confidence intervals as a function of weight. Because we used **covariate.results** (a **marklist** of models) the predictions are averaged over the models in the list and the estimates of precision include model uncertainty. See the help file for detailed information about **covariate.predictions**. Note that the names of the fields in the dataframe must match the names of covariates that you used in the model (e.g., weight).

```
> minmass=min(dipper$weight)
> maxmass=max(dipper$weight)
> mass.values=minmass+(0:30)*(maxmass-minmass)/30
> Phibymass=covariate.predictions(covariate.results,data=data.frame(weight=mass.values),indices=c(1))
# Plot predicted model averaged estimates by weight with pointwise confidence intervals
> plot(Phibymass$estimates$covdata, Phibymass$estimates$estimate,
           type="l",lwd=2,xlab="Mass(kg)",ylab="Survival",ylim=c(0,1))
> lines(Phibymass$estimates$covdata, Phibymass$estimates$lcl,lty=2)
> lines(Phibymass$estimates$covdata, Phibymass$estimates$ucl,lty=2)
```

Now let's consider time-varying individual covariates. **RMark** contains a pre-programmed time-varying covariate which is either age or TSM (time-since-marking) but it is handled via the parameter

structure rather than with an individual covariate with the data. But it is a good example, because it illustrates that the value of time-varying covariates need to be known for each animal at each occasion regardless of whether it was caught or not at the occasion. Thus, the time-varying covariate cannot be one that requires capturing and handling of the animal. Beyond, age an obvious candidate for a time-varying individual covariate for the CJS model is a trap-dependence covariate. The idea here is that animals that were caught on a previous occasion are more likely to be caught on the next occasion. If e_i is the value of the capture history at occasion i, then it becomes the time varying covariate for modeling p_{i+1}. In a CJS model only $p_2, \ldots, p_k$ are estimated for a history with k occasions, so the time varying covariates are $e_1, \ldots, e_{k-1}$ for those parameters. Below with the dipper data we construct a sequence of covariates labeled $td1981, \ldots, td1986$ that contain the capture history entry for the years 1980 to 1985 for each dipper. They are labeled with the 1981 to 1986 suffix because those will be the times for the capture probabilities if we use **begin.time=1980**. Had we not included the assignment of **begin.time**, the times would default to begin at 1, and the variables would have to be named **td2**,. . . ,**td7** to be properly handled by the formula. First we start with a function that creates the trap dependence variable. It was written as a function because it is general and could be used elsewhere; although it would have to be changed if the time intervals between occasions were not 1.

```
> create.td=function(ch,varname="td",begin.time=1)
#
#  Arguments:
#    ch - capture history vector (0/1 values only)
#    varname - prefix for variable name
#    begin.time - time for first occasion
#
#  Value:
#    returns a datframe with trap-dependent variables
#            named varnamet+1,...,varnamet+nocc-1
#            where t is begin.time and nocc is the
#            number of occasions
#
{
# turn vector of capture history strings into a vector of characters
  char.vec=unlist(strsplit(ch,""))
# test to make sure they only contain 0 or 1
  if(!all(char.vec %in% c(0,1)))
    stop("Function only valid for CJS model without missing values")
  else
  {
#    get number of occasions (nocc) and change it into a matrix of numbers
     nocc=nchar(ch[1])
     tdmat=matrix(as.numeric(char.vec),ncol=nocc,byrow=TRUE)
   #   remove the last column which is not used
     tdmat=tdmat[,1:(nocc-1)]
#   turn it into a dataframe and assign the field (column) names
     tdmat=as.data.frame(tdmat)
     names(tdmat)=paste(varname,(begin.time+1):(begin.time+nocc-1),sep="")
     return(tdmat)
  }
}
```

Next we follow with the script that adds the variables to the dipper data and then uses the time-varying covariate in a few models. Note that you only use the prefix (e.g., **td**) in the formula and **RMark** adds the relevant suffix for the parameter.

```
> do.dipper.td=function()
{
# get data and add the td time-varying covariate, process the data
# and create the design data
data(dipper)
dipper=cbind(dipper,create.td(dipper$ch,begin.time=1980))
dipper.processed=process.data(dipper,begin.time=1980)
dipper.ddl=make.design.data(dipper.processed)
# create additional p field adult
dipper.ddl$p$adult=0
dipper.ddl$p$adult[dipper.ddl$p$Age > 1]=1
# define models for Phi
Phi.dot=list(formula=~1)
# define models for p
p.td=list(formula=~td)
p.td.adult=list(formula=~td:adult)
p.td.time=list(formula=~td:time)
p.time.plus.td=list(formula=~time+td)
# create model list
cml=create.model.list("CJS")
# run and return models
return(mark.wrapper(cml,data=dipper.processed,ddl=dipper.ddl))
}
> td.results=do.dipper.td()
```

Rather than focusing on the results, let's look at the design matrices for the models involving the time-varying covariate. In the first model (~**td**), we see that it added each covariate that matched the correct time dependent covariate that matched the parameter for 1981 to 1986 which we can see with the call to PIMS are indices 2 through 7.

```
> td.results[[1]]$design.matrix
                       Phi:(Intercept) p:(Intercept) p:td
 Phi g1 c1980 a0 t1980 "  1"             "0"           "0"
   p g1 c1980 a1 t1981   "0"             "1"           "td1981"
   p g1 c1980 a2 t1982   "0"             "1"           "td1982"
   p g1 c1980 a3 t1983   "0"             "1"           "td1983"
   p g1 c1980 a4 t1984   "0"             "1"           "td1984"
   p g1 c1980 a5 t1985   "0"             "1"           "td1985"
   p g1 c1980 a6 t1986   "0"             "1"           "td1986"

> PIMS(td.results[[1]],"p")
 group = Group 1
      1981 1982 1983 1984 1985 1986
 1980    2    3    4    5    6    7
 1981         3    4    5    6    7
```

```
1982              4    5    6    7
1983                   5    6    7
1984                        6    7
1985                             7
```

Now, if the experiment was one in which the animals were released we might not want to have a trap dependence for the first occasion after the initial release because it might not reflect trap dependence. We can limit the effect to occasions other than the first after release by interacting **td** with the **adult** design covariate (~**adult:td**). Note that parameter 2 does not have the trap-dependence effect and thus **td1981** is not used.

```
> td.results[[2]]$design.matrix
                      Phi:(Intercept) p:(Intercept) p:td:adult
Phi g1 c1980 a0 t1980   "1"             "0"           "0"
  p g1 c1980 a1 t1981   "0"             "1"           "0"
  p g1 c1980 a2 t1982   "0"             "1"           "td1982"
  p g1 c1980 a3 t1983   "0"             "1"           "td1983"
  p g1 c1980 a4 t1984   "0"             "1"           "td1984"
  p g1 c1980 a5 t1985   "0"             "1"           "td1985"
  p g1 c1980 a6 t1986   "0"             "1"           "td1986"

> PIMS(td.results[[2]],"p")
group = Group 1
     1981 1982 1983 1984 1985 1986
1980    2    3    4    5    6    7
1981         2    4    5    6    7
1982              2    5    6    7
1983                   2    6    7
1984                        2    7
1985                             2
```

Now, if you thought that the trap dependence effect might vary by time, you could interact **time** with **td(~time:td)**. Note that here the time effect is only for those caught on the previous occasion. Bit of a strange model without the main effect for time.

```
 > td.results[[3]]$design.matrix
Phi:(Intercept) p:(Intercept) p:td:time1981 p:td:time1982 p:td:time1983 p:td:time1984 p:td:time1985 p:td:time1986
 Phi g1 c1980 a0 t1980   "1"        "0"      "0"        "0"        "0"        "0"        "0"        "0"
   p g1 c1980 a1 t1981   "0"        "1"      "td1981"   "0"        "0"        "0"        "0"        "0"
   p g1 c1980 a2 t1982   "0"        "1"      "0"        "td1982"   "0"        "0"        "0"        "0"
   p g1 c1980 a3 t1983   "0"        "1       "0"        "0"        "td1983"    "0"        "0"       "0"
   p g1 c1980 a4 t1984   "0"        "1"      "0"        "0"        "0"        "td1984"   "0"        "0"
   p g1 c1980 a5 t1985   "0"        "1"      "0"        "0"        "0"        "0"        "td1985"   "0"
   p g1 c1980 a6 t1986   "0"        "1"      "0"        "0"        "0"        "0"        "0"        "td1986"
```

Finally, another model might be one with a time effect and an additive trap dependence effect (~**time+td**).

```
> td.results[[4]]$design.matrix
               Phi:(Intercept) p:(Intercept) p:time1982 p:time1983 p:time1984 p:time1985 p:time1986 p:td
Phi g1 c1980 a0 t1980 "1"          "0"          "0"        "0"        "0"        "0"        "0"     "0"
p g1 c1980 a1 t1981   "0"          "1"          "0"        "0"        "0"        "0"        "0"     "td1981"
p g1 c1980 a2 t1982   "0"          "1"          "1"        "0"        "0"        "0"        "0"     "td1982"
```

```
p g1 c1980 a3 t1983   "0"          "1"          "0"       "1"       "0"       "0"       "0"     "td1983"
p g1 c1980 a4 t1984   "0"          "1"          "0"       "0"       "1"       "0"       "0"     "td1984"
p g1 c1980 a5 t1985   "0"          "1"          "0"       "0"       "0"       "1"       "0"     "td1985"
p g1 c1980 a6 t1986   "0"          "1"          "0"       "0"       "0"       "0"       "1"     "td1986"

> PIMS(td.results[[4]],"p",simplified=F)
 group = Group 1
       1981 1982 1983 1984 1985 1986
  1980   22   23   24   25   26   27
  1981        28   29   30   31   32
  1982             33   34   35   36
  1983                  37   38   39
  1984                       40   41
  1985                            42
```

When **RMark** runs **MARK** with an individual covariate model, it does not standardize the covariates (**MARK** does this on the fly) and **MARK** computes the real parameter estimates using the mean of the covariate value which may not be particularly useful. We'll again demonstrate the use of **covariate.predictions** to show how you can get the predicted values of p with **td=0** and 1 using this final model 4. This is a useful example to show how you limit predictions for covariates to specific parameters because in this case each covariate only applies to one parameter. To do so, the dataframe that is passed to the function should contain a field named **index** which is the parameter index for the non-simplified PIM which is shown above. We want to compute a value of p for **td=0** and **td=1** for each time which can be specified with indices 22 through 27. The following 3 commands create the necessary dataframe as we can tell from the output:

```
> cov.df=data.frame(rbind(diag(rep(1,6)),diag(rep(0,6))))
> names(cov.df)=paste("td",1981:1986,sep="")
> cov.df$index=rep(22:27,2)
>  cov.df
    td1981 td1982 td1983 td1984 td1985 td1986 index
 1       1      0      0      0      0      0    22
 2       0      1      0      0      0      0    23
 3       0      0      1      0      0      0    24
 4       0      0      0      1      0      0    25
 5       0      0      0      0      1      0    26
 6       0      0      0      0      0      1    27
 7       0      0      0      0      0      0    22
 8       0      0      0      0      0      0    23
 9       0      0      0      0      0      0    24
 10      0      0      0      0      0      0    25
 11      0      0      0      0      0      0    26
 12      0      0      0      0      0      0    27
```

The following gets the predicted values and plots them for **td=1** and **td=0** as 2 different lines:

```
> p.est=covariate.predictions(td.results[[4]],data=cov.df)
> plot(1981:1986,p.est$estimates$estimate[1:6],type="b",ylim=c(0,1),xlab="Time",
    ylab="Capture probability",pch=1)
> lines(1981:1986,p.est$estimates$estimate[7:12],type="b",pch=2)
> legend(x=1984,y=.2,legend=c("td=1","td=0"),pch=1:2)
```

C.17. Multi-strata example

So far we have only used the **CJS** model in describing the **RMark** package. Now we switch to giving some examples with some of the other models supported by **RMark** (Table C.1). In general, there is little difference in using any of the models within **RMark** except for differences in the model parameters and some subtle differences due to the model structure. Each of the models in **RMark** comes with an example data set which shows a sample of analyses which often mimic the results in the sample **MARK** **.dbf** for that model.

We start off with the **Multistrata** model because it is a fairly useful model and it follows naturally from a discussion of time-varying covariates. The strata in the **Multistratum** model can be viewed as a time-varying factor variable for each animal except that the stratum (state) for each animal need not be known at each occasion. For the **Multistrata** model we use the **mstrata** data that corresponds to the **mssurv** example that accompanies **MARK**. For the **Multistrata** model there are 3 parameters: ψ (transition), S (survival) and p (capture). There are additional design data for these parameters to accommodate the strata. The strata labels are determined by the alphabetic characters used in the encounter history and need not be **A** to **C** like in this example. Below we show summaries for the design data for ψ and p (S is similar) for this example:

```
> data(mstrata)
> mstrata.processed=process.data(mstrata,model="Multistrata")
> mstrata.ddl=make.design.data(mstrata.processed)
> summary(mstrata.ddl$Psi)

 group  cohort age    time   stratum tostratum     Cohort           Age              Time
 1:36   1:18   0:18   1: 6   A:12    A:12      Min.   :0.0000   Min.   :0.0000   Min.   :0.000
        2:12   1:12   2:12   B:12    B:12      1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:1.000
        3: 6   2: 6   3:18   C:12    C:12      Median :0.5000   Median :0.5000   Median :1.500
                                              Mean   :0.6667   Mean   :0.6667   Mean   :1.333
                                              3rd Qu.:1.0000   3rd Qu.:1.0000   3rd Qu.:2.000
                                              Max.   :2.0000   Max.   :2.0000   Max.   :2.000
                                                  A                B
                                               Min.   :0.0000   Min.   :0.0000
                                               1st Qu.:0.0000   1st Qu.:0.0000
                                               Median :0.0000   Median :0.0000
                                               Mean   :0.3333   Mean   :0.3333
                                               3rd Qu.:1.0000   3rd Qu.:1.0000
                                               Max.   :1.0000   Max.   :1.0000
      C              toA             toB             toC
 Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.   :0.0000
 1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.0000
 Median :0.0000   Median :0.0000   Median :0.0000   Median :0.0000
 Mean   :0.3333   Mean   :0.3333   Mean   :0.3333   Mean   :0.3333
 3rd Qu.:1.0000   3rd Qu.:1.0000   3rd Qu.:1.0000   3rd Qu.:1.0000
 Max.   :1.0000   Max.   :1.0000   Max.   :1.0000   Max.   :1.0000

 summary(mstrata.ddl$p)

 group  cohort age   time  stratum     Cohort           Age             Time            A
 1:18   1:9    1:9   2:3   A:6     Min.   :0.0000   Min.   :1.000   Min.   :0.000   Min.   :0.0000
        2:6    2:6   3:6   B:6     1st Qu.:0.0000   1st Qu.:1.000   1st Qu.:1.000   1st Qu.:0.0000
        3:3    3:3   4:9   C:6     Median :0.5000   Median :1.500   Median :1.500   Median :0.0000
                                   Mean   :0.6667   Mean   :1.667   Mean   :1.333   Mean   :0.3333
```

```
    3rd Qu.:1.0000   3rd Qu.:2.000   3rd Qu.:2.000   3rd Qu.:1.0000
    Max.   :2.0000   Max.   :3.000   Max.   :2.000   Max.   :1.0000
         B                C
Min.   :0.0000   Min.   :0.0000
1st Qu.:0.0000   1st Qu.:0.0000
Median :0.0000   Median :0.0000
Mean   :0.3333   Mean   :0.3333
3rd Qu.:1.0000   3rd Qu.:1.0000
Max.   :1.0000   Max.   :1.0000
```

For all of the parameters, a **stratum** factor variable is included in the design data and a dummy variable (0/1) is included and named with the stratum label. For ψ parameters which describe transition from one stratum to another stratum, there are both **stratum** and **tostratum** factor and dummy variables.

Additional design data can be added with **merge_design.covariates** which can add data based on group and time variables. But if you want to add design data that is specific to particular strata then you'll need to write your own code. You can use the **R** function **merge** or if it is just one or two covariates you can use specific **R** statements that add the covariate as in the following example that adds a distance covariate to the mstrata example.

```
> run.mstrata=function()
{
# Process data
mstrata.processed=process.data(mstrata,model="Multistrata")
# Create default design data
mstrata.ddl=make.design.data(mstrata.processed) # Add distance covariate
mstrata.ddl$Psi$distance=0
mstrata.ddl$Psi$distance[mstrata.ddl$Psi$stratum=="A"&mstrata.ddl$Psi$tostratum=="B"]=12
mstrata.ddl$Psi$distance[mstrata.ddl$Psi$stratum=="A"&mstrata.ddl$Psi$tostratum=="C"]=5
mstrata.ddl$Psi$distance[mstrata.ddl$Psi$stratum=="B"&mstrata.ddl$Psi$tostratum=="C"]=2
mstrata.ddl$Psi$distance[mstrata.ddl$Psi$stratum=="B"&mstrata.ddl$Psi$tostratum=="A"]=12
mstrata.ddl$Psi$distance[mstrata.ddl$Psi$stratum=="C"&mstrata.ddl$Psi$tostratum=="A"]=5
mstrata.ddl$Psi$distance[mstrata.ddl$Psi$stratum=="C"&mstrata.ddl$Psi$tostratum=="B"]=2
# Create formula
Psi.distance=list(formula=~distance)
Psi.distance.time=list(formula=~distance+time)
p.stratum=list(formula=~stratum)
S.stratum=list(formula=~stratum)
model.list=create.model.list("Multistrata")
mstrata.results=mark.wrapper(model.list,data=mstrata.processed,ddl=mstrata.ddl)
return(mstrata.results)
}
> mstrata.results=run.mstrata()
> mstrata.results
```

The code that creates the models in the **MARK** example (**mssurv**) can be found by typing **?mstrata** in **RMark** or can be run by typing **example(mstrata)**. Constructing models for the **Multistrata** parameters is essentially the same as with the CJS model with the exception of ψ which is different due to its unique structure. For each stratum, there are transition parameters to the other strata and the probability of remaining in the stratum is computed by subtraction. Thus, for the **mstrata** example there is a transition from **A** to **B** and **A** to **C** and **A** to **A** is computed by subtraction. The same holds for the other strata. Thus, the **stratum** and **tostratum** factors are not fully crossed by default.

```
> table(mstrata.ddl$Psi[,c("stratum","tostratum")])

        tostratum
stratum   A B C
        A 0 6 6
        B 6 0 6
        C 6 6 0
```

Thus, to specify the interaction of **stratum** and **tostratum** to estimate each ψ parameter without restriction you would use **Psi.s=list(formula=~-1+stratum:tostratum)** and to fit the model with time varying transitions the model the ψ specification would be

```
> Psi.sxtime=list(formula=~-1+stratum:tostratum:time)
```

The transition that is computed by subtraction can be changed with the **subtract.stratum** argument of the **make.design.data** function. For this example the default call is equivalent to:

```
> mstrata.ddl=make.design.data(mstrata.processed,parameters=
              list(Psi=list(subtract.stratum=c("A","B","C"))))
```

but they can also be set such that the same stratum is computed by subtraction for all stratum:

```
> mstrata.ddl=make.design.data(mstrata.processed,parameters=
             list(Psi=list(subtract.stratum=c("B","B","B"))))
```

which does provide fully-crossed **stratum** and **tostratum** factors.

```
> table(mstrata.ddl$Psi[,c("stratum","tostratum")])

        tostratum
stratum A C
      A 6 6
      B 6 6
      C 6 6
```

But, that may not be the best reason for the choice of setting the **subtract.stratum**. Sometimes the choice may be decided based on model convergence. Some choices will yield better convergence if one or more of the ψ parameters is at a boundary.

In some cases, you may want to choose the **subtract.stratum** because you want to specify some ψ values to be set to zero. The easiest way to constrain specific ψ to zero is to delete the design data because that is the default value. However, the ψ that you want to set so zero cannot be computed by subtraction, so you need to set the **subtract.stratum** appropriately. For example, what if you wanted to set **PsiAA=PsiBB=PsiCC=0**? That could be done with the following code for the **mstrata** example:

```
> mstrata.ddl=make.design.data(mstrata.processed,parameters=
             list(Psi=list(subtract.stratum=c("B","A","A"))))
> mstrata.ddl$Psi=mstrata.ddl$Psi[!(mstrata.ddl$Psi$stratum==
             "A"&mstrata.ddl$Psi$tostratum=="A"),]
```

```
> mstrata.ddl$Psi=mstrata.ddl$Psi[!(mstrata.ddl$Psi$stratum==
             "B"&mstrata.ddl$Psi$tostratum=="B"),]
> mstrata.ddl$Psi=mstrata.ddl$Psi[!(mstrata.ddl$Psi$stratum==
             "C"&mstrata.ddl$Psi$tostratum=="C"),]
> mymodel=mark(mstrata.processed,mstrata.ddl)
> summary(mymodel,show.fixed=T)

 Real Parameter Psi
  Stratum:A To:A
   1 2 3
 1 0 0 0
 2   0 0
 3     0

  Stratum:A To:C
           1         2         3
 1 0.5014851 0.5014851 0.5014851
 2           0.5014851 0.5014851
 3                     0.5014851

  Stratum:B To:B
   1 2 3
 1 0 0 0
 2   0 0
 3     0

  Stratum:B To:C
           1         2         3
 1 0.5014063 0.5014063 0.5014063
 2           0.5014063 0.5014063
 3                     0.5014063

  Stratum:C To:B
           1         2         3
 1 0.4999394 0.4999394 0.4999394
 2           0.4999394 0.4999394
 3                     0.4999394


  Stratum:C To:C
   1 2 3
 1 0 0 0
 2   0 0
 3     0
```

In other cases, the choice may be determined based on the ability to restrict equality for specific transitions. For example, if you had an example with 2 strata (**A** & **B**) and you wanted to set **PsiAB=PsiBB** you could do that by setting **subtract.stratum=c("A","A")** and fitting the **intercept(constant)** model for ψ. That gets more difficult with 3 or more strata. However, sometimes you can use design data to

create constraints. For example, with the **mstrata** data, if you wanted to fit **PsiAB=PsiBA=PsiCA** and **PsiAC=PsiBC=PsiCC**, then you could use the following **subtract.stratum** and formula:

```
> data(mstrata)
> mstrata.processed=process.data(mstrata,model="Multistrata")
> mstrata.ddl=make.design.data(mstrata.processed,parameters
          =list(Psi=list(subtract.stratum=c("A","B","B"))))
> mark(mstrata.processed,mstrata.ddl,model.parameters=list(Psi=list(formula=~toC)))

<...>

  Real Parameter Psi
   Stratum:A To:B
             1         2         3
  1 0.2175964 0.2175964 0.2175964
  2           0.2175964 0.2175964
  3                     0.2175964

   Stratum:A To:C
             1         2         3
  1 0.2132953 0.2132953 0.2132953
  2           0.2132953 0.2132953
  3                     0.2132953

   Stratum:B To:A
             1         2         3
  1 0.2175964 0.2175964 0.2175964
  2           0.2175964 0.2175964
  3                     0.2175964

   Stratum:B To:C
             1         2         3
  1 0.2132953 0.2132953 0.2132953
  2           0.2132953 0.2132953
  3                     0.2132953

   Stratum:C To:A
             1         2         3
  1 0.2175964 0.2175964 0.2175964
  2           0.2175964 0.2175964
  3                     0.2175964

   Stratum:C To:C
             1         2         3
  1 0.2132953 0.2132953 0.2132953
  2           0.2132953 0.2132953
  3                     0.2132953
```

Now because the other parameters are computed by subtraction, it also set **PsiAA=PsiBB=PsiCB**.

What if you only wanted to set **PsiBC=PsiCC**? First, you could define a dummy variable **bc.toC** that was 1 for strata **B** and **C** for the transitions to **C** as follows:

```
> mstrata.ddl$Psi$bc.toC=0
> mstrata.ddl $Psi$bc.toC [mstrata.ddl $Psi$stratum%in%c("B","C")&
         mstrata.ddl $Psi$tostratum=="C"]=1
```

Then using the same **subtract.stratum** values you would naturally try:

```
 mark(mstrata.processed,mstrata.ddl,model.parameters=list(Psi=list(formula=~bc.toC)))

<...>

 Real Parameter Psi
  Stratum:A To:B
          1        2        3
 1 0.222929 0.222929 0.222929
 2          0.222929 0.222929
 3                   0.222929

  Stratum:A To:C
          1        2        3
 1 0.222929 0.222929 0.222929
 2          0.222929 0.222929
 3                   0.222929

  Stratum:B To:A
           1         2         3
 1 0.1731952 0.1731952 0.1731952
 2           0.1731952 0.1731952
 3                     0.1731952

  Stratum:B To:C
           1         2         3
 1 0.3962874 0.3962874 0.3962874
 2           0.3962874 0.3962874
 3                     0.3962874

  Stratum:C To:A
           1         2         3
 1 0.1731952 0.1731952 0.1731952
 2           0.1731952 0.1731952
 3                     0.1731952

  Stratum:C To:C
           1         2         3
 1 0.3962874 0.3962874 0.3962874
 2           0.3962874 0.3962874
 3                     0.3962874
```

You might have been expecting that **PsiAB=PsiBA=PsiCA=PsiAC** but now we get **PsiAB=PsiAC** and **PsiCA=PsiBA** but the pairs differ. From the design matrix with just 2 columns you would not expect to get 3 different estimates. To understand what is happening you need to understand the **mlogit** link and how it relates to the β's. Below are the equations for each of the above ψ parameters using β_0 as the intercept and β_1 as the value for **bc.toC**:

$$\psi^{AB} = \psi^{AC} = \frac{\exp(\beta_0)}{1 + \exp(\beta_0) + \exp(\beta_0)}$$

$$\psi^{BA} = \psi^{CA} = \frac{\exp(\beta_0)}{1 + \exp(\beta_0) + \exp(\beta_0 + \beta_1)}$$

$$\psi^{BC} = \psi^{CC} = \frac{\exp(\beta_0 + \beta_1)}{1 + \exp(\beta_0) + \exp(\beta_0 + \beta_1)}$$

Due to the way the **mlogit** link is constructed, if you restrict parameters across a partial subset of the strata, then it may not be possible to construct the model you want. The solution is to change the link function to **logit** as shown below.

```
> mark(mstrata.processed,mstrata.ddl,model.parameters
        =list(Psi=list(formula=~bc.toC,link="logit")))

<...>

Real Parameter Psi
 Stratum:A To:B
          1         2         3
1 0.1993645 0.1993645 0.1993645
2           0.1993645 0.1993645
3                     0.1993645

 Stratum:A To:C
          1         2         3
1 0.1993645 0.1993645 0.1993645
2           0.1993645 0.1993645
3                     0.1993645

 Stratum:B To:A
          1         2         3
1 0.1993645 0.1993645 0.1993645
2           0.1993645 0.1993645
3                     0.1993645

 Stratum:B To:C
         1         2         3
1 0.3990723 0.3990723 0.3990723
2           0.3990723 0.3990723
3                     0.3990723
```

```
 Stratum:C To:A
          1         2         3
1 0.1993645 0.1993645 0.1993645
2           0.1993645 0.1993645
3                     0.1993645

 Stratum:C To:C
          1         2         3
1 0.3990723 0.3990723 0.3990723
2           0.3990723 0.3990723
3                     0.3990723
```

Why not use the **logit** link all of the time? You can do that but the **mlogit** link was chosen as the default for **RMark** because it provides a natural constraint to make sure the real values sum to 1. If you choose to use the **logit** link, then just beware that **MARK** enforces the constraint by penalizing the likelihood and that may not be as stable numerically. Clearly, to build some models you may be required to use the **logit** link. Make sure to look at the penalty value in the **MARK** output to make sure that the penalty value is 0. The **logit** link does have the additional advantage that the PIMS can be simplified whereas they cannot be simplified with the **mlogit** link. But, beware that some of the **RMark** code for computation on the results from **MARK** has been written specifically for the **mlogit** link.

The ψ estimates for the **subtract.stratum** are not given by **MARK**. Obviously, computing the point estimate is simple by summing the other values and subtracting from 1. However, computing the standard error and confidence interval is more tedious. To avoid doing this by hand, the function **TransitionMatrix** was created to compute each real parameter, standard error and confidence interval. At present, it only works if you use the **mlogit** link with ψ. See the help file for that function and **get.real** for more details. The following will run the example code for **mstrata** and then compute the transition matix.

```
> example(mstrata)
> Psilist=get.real(mstrata.results[[1]],"Psi",vcv=T)
> Psivalues=Psilist$estimates

> TransitionMatrix(Psivalues[Psivalues$time==1,],vcv.real=Psilist$vcv.real)

 $TransitionMat
          A         B         C
A 0.6020772 0.1993450 0.1985778
B 0.1993452 0.6020771 0.1985777
C 0.2003789 0.2003787 0.5992424

> $se.TransitionMat

            A          B          C
A 0.01863979 0.01412477 0.01413614
B 0.01412478 0.01863984 0.01413616
C 0.01422173 0.01422172 0.01871430

> $lcl.TransitionMat
```

```
           A         B         C
A 0.5650384 0.1730952 0.1723155
B 0.1730954 0.5650382 0.1723153
C 0.1739486 0.1739485 0.5620711

> $ucl.TransitionMat

           A         B         C
A 0.6379827 0.2284757 0.2277414
B 0.2284760 0.6379827 0.2277414
C 0.2297083 0.2297081 0.6353057
```

C.18. Nest survival example

The nest survival model is quite different than most of the other models in **MARK** because it is **not** based on an encounter history. At present, neither **convert.inp** nor **import.chdata** will handle data entry for nest survival data. The data must be imported into an **R** dataframe and certain fields must be included with specific names. Two examples are provided in **RMark**. The killdeer example is the data that accompanies **MARK** and the mallard example provided by Jay Rotella is documented in

> Rotella, J. J., S. J. Dinsmore, T. L. Shaffer. 2004. Modeling nest-survival data: a comparison of recently developed methods that can be implemented in **MARK** and **SAS**. *Animal Biodiversity and Conservation* **27**:187-204.

The dataframe must contain the following variables with these names:

- **FirstFound**: day the nest was first found
- **LastPresent**: last day that a chick was present in the nest
- **LastChecked**: last day the nest was checked
- **Fate**: fate of the nest; 0=hatch and 1=depredated

It can also contain a field **Freq** which is the frequency of nests with this data. If it is always 1 then it is not needed. The dataframe can also contain any number of other covariate or identifier fields. If your dataframe contains a variable **AgeDay1**, which is the age of the nest on the first occasion then you can use a variable called **NestAge** in the formula which will create a set of time-dependent covariates named **NestAge1**,**NestAge2**,. . . ,**NestAge(nocc-1)** which will provide a way to incorporate the age of the nest in the model. The use of **AgeDay1** and **NestAge** was added because the age covariate in the design data for the parameter *S* (survival) assumes all nests are the same age and is not particularly useful. This effect could be incorporated by using the **add()** function in the design matrix but **RMark** does not have any capability for doing that and it is easier to create a time-dependent covariate to do the same thing.

The file **killdeer.inp** and **mallard.txt** come with **RMark**. The code below provides examples for importing and setting up nest survival data for **RMark**. Modify the path to **Rmark** as needed.

```
# EXAMPLE CODE FOR CONVERSION OF .INP TO NECESSARY DATA STRUCTURE
# read in killdeer.inp file
```

```
> killdeer=scan("C:/Program Files/R/R-2.6.0/library/RMark/data/killdeer.inp",
        what="character",sep="\n")
# strip out ; and write out all but first 2 lines which contain comments
> write(sub(";","",killdeer[3:20]),"killdeer.txt")
# read in as a dataframe from tab-delimited text file and assign names
> killdeer=read.table("killdeer.txt")
> names(killdeer)=c("id","FirstFound","LastPresent","LastChecked",
                              "Fate","Freq")
# Read in data, which are in a simple text file that
# looks like a MARK input file but (1) with no comments or semicolons and
# (2) with a 1st row that contains column labels
> mallard=read.table("C:/Program Files/R/R-2.6.0/library/RMark/data/mallard.txt",
  header=TRUE)
```

The help files for **killdeer** and **mallard** provide example code for analysis of nest survival data. In particular, the script in the **mallard** help file is a nice example constructed by Jay Rotella. It demonstrates the benefits of **RMark** and provides a useful model for scripting an entire analysis from model building to prediction and plotting. It uses an alternative approach with **find.covariates**, **fill.covariates** and **compute.real** functions which were created before **covariate.predictions**. We have extended this example further here to include a 3-D plot:

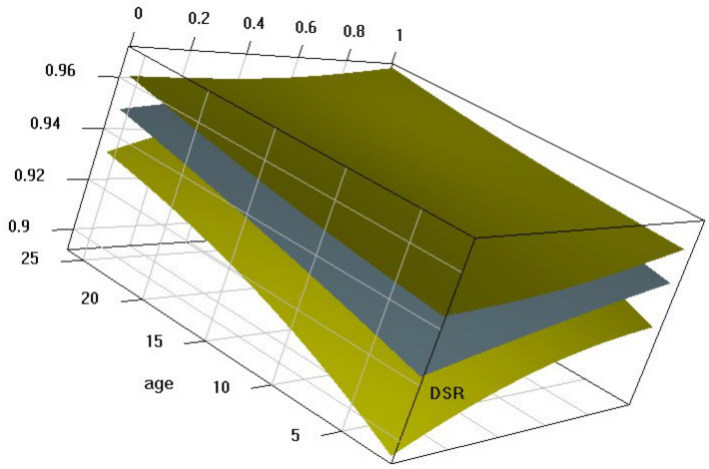
```
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~#
# Example of use of RMark for modeling nest survival data - Mallard nests example        #
# The example runs the 9 models that are used in the Nest Survival chapter               #
# of the Gentle Introduction to MARK and that appear in Table 3 (page 198) of            #
# Rotella, J.J., S. J. Dinsmore, T.L. Shaffer.  2004.  Modeling nest-survival data:      #
#   a comparison of recently developed methods that can be implemented in MARK and SAS.  #
#   Animal Biodiversity and Conservation 27:187-204.                                     #
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~#
> data(mallard)

# Treat dummy variables for habitat types as factors
> mallard$Native=factor(mallard$Native)
> mallard$Planted=factor(mallard$Planted)
> mallard$Wetland=factor(mallard$Wetland)
> mallard$Roadside=factor(mallard$Roadside)

# Examine a summary of the dataset
> summary(mallard)

# Write a function for evaluating a set of competing models
> run.mallard=function()
{
# 1. A model of constant daily survival rate (DSR)
Dot=mark(mallard,nocc=90,model="Nest",model.parameters=list(S=list(formula=~1)))

# 2. DSR varies by habitat type - treats habitats as factors
#  and the output provides S-hats for each habitat type
Hab=mark(mallard,nocc=90,model="Nest",
  model.parameters=list(S=list(formula=~Native+Planted+Wetland)),
  groups=c("Native","Planted","Wetland"))

# 3. DSR varies with vegetation thickness (Robel reading)
```

```
# Note: coefficients are estimated using the actual covariate
# values. They are not based on standardized covariate values.
Robel=mark(mallard,nocc=90,model="Nest",
  model.parameters=list(S=list(formula=~Robel)))

# 4. DSR varies with the amount of native vegetation in the surrounding area
# Note: coefficients are estimated using the actual covariate
# values. They are not based on standardized covariate values.
PpnGr=mark(mallard,nocc=90,model="Nest",model.parameters=list(S=list(formula=~PpnGrass)))

# 5. DSR follows a trend through time
TimeTrend=mark(mallard,nocc=90,model="Nest",model.parameters=list(S=list(formula=~Time)))

# 6. DSR varies with nest age
Age=mark(mallard,nocc=90,model="Nest",model.parameters=list(S=list(formula=~NestAge)))

# 7. DSR varies with nest age & habitat type
AgeHab=mark(mallard,nocc=90,model="Nest",
  model.parameters=list(S=list(formula=~NestAge+Native+Planted+Wetland)),
  groups=c("Native","Planted","Wetland"))

# 8. DSR varies with nest age & vegetation thickness
AgeRobel=mark(mallard,nocc=90,model="Nest",
  model.parameters=list(S=list(formula=~NestAge+Robel)))

# 9. DSR varies with nest age & amount of native vegetation in surrounding area
AgePpnGrass=mark(mallard,nocc=90,model="Nest",
  model.parameters=list(S=list(formula=~NestAge+PpnGrass)))

#
# Return model table and list of models
#
return(collect.models() )
}

> mallard.results=run.mallard()  # This runs the 9 models above and takes a minute or 2

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~#
# Examine table of model-selection results #
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~#
> mallard.results                                    # print model-selection table to screen
> options(width=100)                                 # set page width to 100 characters
> sink("results.table.txt")                          # capture screen output to file
> print.marklist(mallard.results)                    # send output to file
> sink()                                             # return output to screen
> system("notepad results.table.txt",invisible=FALSE)    # view results in notepad

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~#
# Examine output for constant DSR model #
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~#
> mallard.results$Dot                          # print MARK output to designated text editor
> mallard.results$Dot$results$beta             # view estimated beta for model in R
> mallard.results$Dot$results$real             # view estimated DSR estimate in R
```

```
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~#
# Examine output for 'DSR by habitat' model #
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~#
> mallard.results$Hab                          # print MARK output to designated text editor
> mallard.results$Hab$design.matrix            # view the design matrix that was used
> mallard.results$Hab$results$beta             # view estimated beta for model in R
> mallard.results$Hab$results$beta.vcv         # view variance-covariance matrix for beta's
> mallard.results$Hab$results$real             # view the estimates of Daily Survival Rate

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~#
# Examine output for best model #
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~#
> mallard.results$AgePpnGrass                  # print MARK output to designated text editor
> mallard.results$AgePpnGrass$results$beta     # view estimated beta's in R
> mallard.results$AgePpnGrass$results$beta.vcv # view estimated var-cov matrix in R

# To obtain estimates of DSR for various values of 'NestAge' and 'PpnGrass'
#   some work additional work is needed.
# First, a simpler name for the object containing the preferred model results
> AgePpnGrass=mallard.results$AgePpnGrass
# Build design matrix with ages and ppn grass values of interest
> fc <- find.covariates(AgePpnGrass,mallard)
# iterate through sequence of ages and proportion grassland
#     values to build prediction surfaces
> seq.ages <- seq(2, 26, by=2)
> seq.ppn <- seq(0.01,0.99,length=89)
> point <- matrix(nrow=89, ncol=length(seq.ages))
> lower <- matrix(nrow=89, ncol=length(seq.ages))
> upper <- matrix(nrow=89, ncol=length(seq.ages))
> colnum <- 0
> for (iage in seq.ages) {
  fc$value[1:89]=iage                       # assign sequential age
  colnum <- colnum + 1
  fc$value[fc$var=="PpnGrass"]=seq.ppn # assign range of values to PpnGrass
  design=fill.covariates(AgePpnGrass,fc) # fill design matrix with values
  point[,colnum] <- compute.real(AgePpnGrass,design=design)[,"estimate"]
  lower[,colnum] <- compute.real(AgePpnGrass,design=design)[,"lcl"]
  upper[,colnum] <- compute.real(AgePpnGrass,design=design)[,"ucl"]
  }
#  Predicted surfaces shown in a window that can be rotated and zoomed by user
#     left mouse button=rotate, right mouse=zoom
> library(rgl)
> open3d()
> bg3d("white")
> material3d(col="black")
> persp3d(seq.ppn, seq.ages, point, aspect=c(1, 1, 0.5), col = "lightblue",
        xlab = "grass", ylab = "age", zlab = "DSR", zlim=range(c(upper,lower)),
        main="Daily survival rate (with CI)", sub="for model 'age and proportion grassland'")

> persp3d(seq.ppn, seq.ages, upper, aspect=c(1, 1, 0.5), col = "yellow", add=TRUE)

> persp3d(seq.ppn, seq.ages, lower, aspect=c(1, 1, 0.5), col = "yellow", add=TRUE)
```

```
> grid3d(c("x","y+","z"))
#  see rgl.snapshot(file="snapshot.png")  for creating png image of generated surfaces
```

The script fits 9 models to the data, then goes on to examine the best model and produce predicted point estimates and confidence intervals for a grid of values for the covariates (proportion native vegetation, and nest age). The 3 surfaces are then graphed, using a dynamic graphics library available in **R**, named **rgl**. This permits viewing of the surface by rotating and tilting, then capturing the most illuminating view of the surfaces (shown below).



C.19. Occupancy examples

The occupancy models are more similar to the encounter history models in **MARK** than the nest survival example but the histories relate to sites rather than animals and the values are presence(1)/absence(0) or counts of animals at a site. At present there are 13 different occupancy models in **MARK** that are supported by **RMark**:

Occupancy, OccupHet, RDOccupEG, RDOccupPE, RDOccupPG, RDOccupHetEG, RDOccupHetPE, RDOccupHetPG, OccupRNPoisson, OccupRNNegBin, OccupRPoisson, OccupRNegBin, MSOccupancy.

Het means it uses the Pledger mixture for the detection probability model and those with RD are the robust design models. The 2 letter designations for the RD models are shorthand for the parameters that are estimated: 1) for EG, ψ, ϵ, and γ are estimated, 2) for PE, γ is dropped and 3) for PG, ϵ is dropped. For the latter 2 models, ψ can be estimated for each primary occasion.

The last 5 models include the Royle/Nichols count (**OccupRPoisson, OccupRNegBin**) and presence (**OccupRNPoisson, OccupRNNegBin**) models with Poisson and Negative Bionomial versions and the multi-state occupancy model (MSOccupancy). Each of the models and parameters are shown in Table C.1.

Example datasets are provided with **RMark** for each of the models. See the example datasets **salamander** and **weta** for Occupancy and OccupHet, **Donovan.7** for an example of OccupRNPoisson and OccupRNNegBin, **Donovan.8** for an example of OccupRPoisson and OccupRNegBin, **RDSalamander** for an example of the robust design models and **NicholsMSOccupancy** for an example of MSOccupancy. Here we provide more in-depth description and examples for the Occupancy and MSOccupancy models.

Imagine a scenario in which you wanted to model species-habitat dependent occupancy with detection probability dependent on effort which varied by occasion and site. An example dataset (**mydata.txt**) might look as follows in a tab-delimited file with the variable names in the first row:

```
ch freq Species Habitat Effort1 Effort2 Effort3 Effort4 Effort5
00111 1 LGB Forest 5 2 14 2 5
00111 1 LGB Forest 5 3 16 3 5
10011 1 LGB Forest 4 2 11 2 4
10110 1 LGB Forest 5 3 15 3 5
00.00 1 LBB Forest 5 3 16 3 5
00000 1 LBB Forest 6 3 19 3 6
00010 1 LBB Forest 3 1 8 1 3
000.0 1 LBB Forest 5 3 16 3 5
00000 1 LBB Forest 4 2 13 2 4
00111 1 LBB Forest 4 2 12 2 4
00000 1 LBB Forest 5 2 14 2 5
00111 1 LBB Forest 3 2 10 2 3
00000 1 LBB Grassland 4 2 11 2 4
00000 1 LBB Grassland 3 2 10 2 3
00000 1 LBB Grassland 4 2 12 2 4
00000 1 LBB Grassland 2 1 6 1 2
00100 1 LGB Grassland 5 2 15 2 5
00100 1 LGB Grassland 4 2 13 2 4
```

Note the use of "." for cases in which as site was not visited. The file could be imported with the command

```
> CovOccup=import.chdata("mydata.txt",field.types=c("n","f","f","n","n","n","n","n"))
```

which denotes that **freq** is a numeric field (**"n"**), **Species** and **Habitat** are factor variables (**"f"**) and **Effort1** → **Effort5** are numeric. A **field.type** is not given for ch because it is always assumed to be a character string and *must always be the first field in each record*. The naming of **Effort1**→ **Effort5** assumes that you'll use the default of **begin.time=1** because it is a time-varying covariate. An example run in **RMark** could be as follows:

```
# fit an additive Species+Habitat Psi model and Effort model for p
> mark(CovOccup,model="Occupancy",group=c("Species","Habitat"),
    model.parameters=list(Psi=list(formula=~Species+Habitat),p=list(formula=~Effort)))
```

Time-varying covariates are particularly useful for occupancy models because they relate to the site so they should always be known for each time (occasion). In most cases the time-varying covariates would be values like effort, weather, number of observers for detection probability but they could also be used for ψ. If the time-varying covariates are factor variables then you will need to create a dummy variable for the levels. For example, let's say you had 2 different observers doing the site visits and you thought that one observer might be more diligent than the other at searching. A factor variable with k levels requires $k-1$ dummy variables. In this case, $k=2$, so we only need one dummy variable that we'll assign to **observer2**. If the site was visited by **observer2** on occasion j then the variable **observer2j** would be assigned a 1 and a 0 otherwise. You would need **observer21** → **observer2n** if you had n visits (occasions) to each site.

Some example data might look as follows in which observer 2 visited site 1 on occasions 2 and 4, and site 2 on occasions 1 and 5:

```
ch freq Species Habitat Observer21 Observer22 Observer23 Observer24 Observer25
00111 1 LGB Forest 0 1 0 1 0
00111 1 LGB Forest 1 0 0 0 1
```

The variable **Observer2** could be used in place of **Effort** in the example formulas shown above. If the factor covariate had more levels then you would have to add additional variables and they would also be included in the formula. You can think of those variables as you would columns in a design matrix except that their values would vary across sites/occasions. However, if you get many levels of the factor variable and many site visits, it is rather onerous to create all of those variables. Therefore, with a slightly clever usage of **model.matrix**, we created and added a function called **make.time.factor** to convert time-varying factor variables into a set of time-varying dummy variables. We demonstrate its usage with the **weta** dataset which is analyzed in MacKenzie *et al*. (2006) on pg 116-122, and which accompanies the program **PRESENCE** (which can be obtained from **http://www.mbr-pwrc.usgs.gov/software/presence.html**). From their Excel file we constructed the text file **weta.txt** which is in the data subdirectory of the **RMark** package. The following is the first few lines of the data:

```
ch        Browse      Obs1      Obs2      Obs3      Obs4      Obs5
0000.        1        1         3         2         3         .
0000.        1        1         3         2         3         .
0001.        1        1         3         2         3         .
0000.        0        1         3         2         3         .
0000.        1        1         3         2         3         .
```

The variables **Obs1** → **Obs5** contain the number of the observer that conducted the visit 1 to 5 and a "." if the site was not visited. Each variable is read in as a factor variable with the following call to **import.chdata**:

```
> weta=import.chdata("weta.txt",field.types=c(rep("f",6)))
```

Each observer factor has 3 levels: 1,2,3 (excluding "."). To construct, dummy variables from a factor variable, one level is chosen as an intercept (observer 3 in this case) and you need $k - 1$ (3-1=2) dummy variables for each time (visit). As with the time-varying effort variable above, these time-varying covariates have a suffix that creates a linkage with the time (visit). The following call to **make.time.factor**, creates those dummy variables (**Obs11**,. . . ,**Obs15**,**Obs21**,. . . ,**Obs25**) from 1 → 5 and replaces them in the data frame:

```
> summary(weta)

      ch            Browse Obs1   Obs2   Obs3   Obs4   Obs5
 Length:72          0:37   .:19   .:15   .:12   .:24   .:28
 Class :character   1:35   1:21   1:17   1:20   1:18   1:15
 Mode  :character          2:12   2:20   2:20   2:15   2:14
                           3:20   3:20   3:20   3:15   3:15

> weta=make.time.factor(weta,"Obs",1:5,intercept=3)
> summary(weta)
```

```
      ch            Browse     Obs11            Obs21            Obs12            Obs22
 Length:72          0:37   Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.   :0.0000
 Class :character   1:35   1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.0000
 Mode  :character          Median :0.0000   Median :0.0000   Median :0.0000   Median :0.0000
                           Mean   :0.2917   Mean   :0.1667   Mean   :0.2361   Mean   :0.2778
                           3rd Qu.:1.0000   3rd Qu.:0.0000   3rd Qu.:0.0000   3rd Qu.:1.0000
                           Max.   :1.0000   Max.   :1.0000   Max.   :1.0000   Max.   :1.0000
     Obs13            Obs23            Obs14          Obs24            Obs15
 Min.   :0.0000   Min.   :0.0000   Min.   :0.00   Min.   :0.0000   Min.   :0.0000
 1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.00   1st Qu.:0.0000   1st Qu.:0.0000
 Median :0.0000   Median :0.0000   Median :0.00   Median :0.0000   Median :0.0000
 Mean   :0.2778   Mean   :0.2778   Mean   :0.25   Mean   :0.2083   Mean   :0.2083
 3rd Qu.:1.0000   3rd Qu.:1.0000   3rd Qu.:0.25   3rd Qu.:0.0000   3rd Qu.:0.0000
 Max.   :1.0000   Max.   :1.0000   Max.   :1.00   Max.   :1.0000   Max.   :1.0000
     Obs25
 Min.   :0.0000
 1st Qu.:0.0000
 Median :0.0000
 Mean   :0.1944
 3rd Qu.:0.0000
 Max.   :1.0000
```

Then the phrase **Obs1+Obs2** can be used in a formula to include an observer effect. See the help for **weta** or use **example(weta)** to explore the example further.

If by chance or design, a single observer was used for all sites on each occasion but the observers varied with occasion, then it isn't necessary to use a time-varying covariate and you can simply assign the observer level to the design data and use it in a model as follows (do not expect reasonable results with just 2 lines of data):

```
mydata.txt contents:

ch freq Species Habitat
00111 1 LGB Forest
00111 1 LGB Forest

> CovOccup=import.chdata("mydata.txt",field.types=c("n",rep("f",2)))
> CovOccup.process=process.data(CovOccup,model="Occupancy")
> CovOccup.ddl=make.design.data(CovOccup.process)
> CovOccup.ddl$p=merge_design.covariates(CovOccup.ddl$p,
      df=data.frame(time=1:5,observer=c("2","3","1","2","3")))

> CovOccup.ddl$p
  group age time Age Time observer
1     1   0    1   0    0        2
2     1   1    2   1    1        3
3     1   2    3   2    2        1
4     1   3    4   3    3        2
5     1   4    5   4    4        3

> mark(CovOccup.process,CovOccup.ddl,model.parameters=list(p=list(formula=~observer)))
```

For this simple case, we could have simply used an assignment statement to create observer, but had there been groups of sites, then **merge_design.covariates** is a better approach. Also, note the use of quote marks for the observer value to create a factor variable. If the quotes had not been used, the variable would have been improperly treated as a numeric variable (i.e., observer 2 effect is twice observer 1 effect).

Next, we'll move onto an example analysis of the **MSOccupancy** model that can be compared to the results in the recent manuscript by Nichols *et al*. (2007). We chose this model to demonstrate the relatively rare situation where parameters can share columns in the design matrix. As described in section C.3, most parameters do not share columns in the design matrix and for the exceptions, the argument **share=TRUE** or **FALSE** was added to the formula for the dominant parameter which was specified arbitrarily (p_1 in this case). In this case, p_1 and p_2 are detection probabilities for states 1 and 2 and often we will want to fit models where these parameters are equated or share covariate values. When **share=TRUE**, only a formula for the dominant parameter is specified but if **share=FALSE**, then a formula for both parameters are expected.

Nichols *et al*. (2007) specified 4 different models for detection probability: 1) variation in time but not by state $(1/2)$, 2) time-invariant and $p_1 = p_2$, 3) time-invariant but $p1 \neq p_2$, and 4) time and state varying. For the first 2 models, p_1 and p_2 share columns in the design matrix and in the last 2 they do not share columns. The parameter specifications for these models are:

1. **p1=list(formula=~time,share=TRUE)**
2. **p1=list(formula=~1,share=TRUE)**
3. **p1=list(formula=~1,share=FALSE)**
   p2=list(formula=~1)
4. **p1=list(formula=~time,share=FALSE)**
   p2=list(formula=~time)

The script with these formulas that replicates the results of Nichols *et al*. (2007) is shown below. Note that there are some very minor differences in the AIC values which may be due to rounding.

```
# To create the data file use:
# NicholsMSOccupancy=convert.inp("NicholsMSOccupancy.inp")
#
# Create a function to fit the 12 models in Nichols et al (2007).
> do.MSOccupancy=function()
{
#  Get the data
   data(NicholsMSOccupancy)
# Define the models; default of Psi1=~1 and Psi2=~1 is assumed
# p varies by time but p1t=p2t
   p1.p2equal.by.time=list(formula=~time,share=TRUE)
# time-invariant p p1t=p2t=p1=p2
   p1.p2equal.dot=list(formula=~1,share=TRUE)
#time-invariant p1 not = p2
   p1.p2.different.dot=list(p1=list(formula=~1,share=FALSE),p2=list(formula=~1))
# time-varying p1t and p2t
   p1.p2.different.time=list(p1=list(formula=~time,share=FALSE),p2=list(formula=~time))
# delta2 model with one rate for times 1-2 and another for times 3-5; delta2 defined below
   Delta.delta2=list(formula=~delta2)
   Delta.dot=list(formula=~1)  # constant delta
```

```
   Delta.time=list(formula=~time) # time-varying delta
# Process the data for the MSOccupancy model
   NicholsMS.proc=process.data(NicholsMSOccupancy,model="MSOccupancy")
# Create the default design data
   NicholsMS.ddl=make.design.data(NicholsMS.proc)
# Add a field for the Delta design data called delta2.
# It is a factor variable with 2 levels: times 1-2, and times 3-5.
   NicholsMS.ddl=add.design.data(NicholsMS.proc,NicholsMS.ddl,
     "Delta",type="time",bins=c(0,2,5),name="delta2")
# Create a list using the 4 p models and 3 delta models (12 models total)
   cml=create.model.list("MSOccupancy")
# Fit each model in the list and return the results
   return(mark.wrapper(cml,data=NicholsMS.proc,ddl=NicholsMS.ddl))
}
# Call the function to fit the models and store it in MSOccupancy.results
> MSOccupancy.results=do.MSOccupancy()
# Print the model table for the results
> print(MSOccupancy.results)
# Adjust model selection by setting chat=1.74 used in the paper
> MSOccupancy.results=adjust.chat(chat=1.74,MSOccupancy.results)
# Print the adjusted model selection results table
> print(MSOccupancy.results)
```

The script also illustrates how to use **add.design.data** to accommodate their use of **delta2** and it also shows a feature that was recently added to **create.model.list** and **mark.wrapper**. These functions were originally designed to construct all possible combinations of parameter specifications. However, this example shows how those functions can now cope with lists that include more than one parameter specification. For example, the p_2 formula here will only be paired with the p_1 formula contained in the list

```
> p1.p2.different.time=list(p1=list(formula=~time,share=FALSE),p2=list(formula=~time))
```

and not with other p_1 formula where it would not be appropriate (i.e., **share=TRUE**).

C.20. Known fate example

The known fate model (**model="Known"**) has only one parameter S for survival. It uses the LD format for data entry. Below is the data description from the **MARK** help file (see also Chapters 2 and 16):

> *"The data coding for the known fate model requires a* **1** *in the* **L** *part of the encounter history for every occasion that the animal is alive at the start of the interval and its fate is known through the interval. A* **10** *means the animal lived through the interval, and a* **11** *means the animal died during the interval. There is no code of* **01** *allowed in the known fate model – this means that the animal was not alive at the start of the interval, so could not have died. To censor an animal for an interval where you don't know what was happening, use the* **00** *code. Thus, the encounter history* **00101000101100** *means that the animal lived through intervals 2 and 3, was censored for interval 4, lived through 5, and died in interval 6."*

We will use the **Blackduck** known-fate example that accompanies **MARK** and **RMark** and using **example(Blackduck)** to run some of the models that are in the **Blckduck.dbf/.fpt** files with **MARK**. Our main focus in this section will be to show some of the flexibility in **RMark** for handling time and age that can be useful with analysis of known fate data that span several years and ages with possibly overlapping time intervals. We will use (and likely abuse) the **Blackduck** example by arbitrarily dividing the data in half with the first half initiated 1 Jan 2000 and the second half in 1 Jan 2001. We'll also pretend

that each occasion represents 0.25 years, so the 8 occasions represent 2 years. Thus, the data for 2000 will span 1 Jan 2000 - 31 Dec 2001 and the data for 2001 will span 1 Jan 2001 to 31 Dec 2002. Also, weŠll have birds that were initially age 0 and age 1, so they can range in ages from 0 to 3 throughout the course of the data. The following code retrieves and defines the data with **year** and **BirdAge** changed to factor variables show they can be used to define groups:

```
data(Blackduck)
Blackduck$year=c(rep(2000,24),rep(2001,24))
Blackduck$year=factor(Blackduck$year)
Blackduck$BirdAge=factor(Blackduck$BirdAge)
```

Next we want to process the data and set the parameters to define the group, time and age structure that we have created. The group structure is defined with **groups=c("year","BirdAge")** and **age.var=2** specifies that the second group variable (**"BirdAge"**) should be treated as the factor for variable for defining initial ages. The age of the animals at the time of release for the 2 age groups were specified with **initial.age=c(0,1)**. They could have been different from the values of **BirdAge**. Next, the time intervals between occasions are set at 0.25 for each occasion. Finally, the initial time of release for each of the 4 groups is specified with **begin.time=c(2000,2001,2000,2001)**. Had the ordering of the group variables been swapped then it would have been specified as **begin.time=c(2000,2000,2001,2001)** with **age.var=1**.

```
> Blackduck.process=process.data(Blackduck,model="Known",groups=c("year","BirdAge"), age.var=2,
  initial.age=c(0,1), time.intervals=rep(.25,8), begin.time=c(2000,2001,2000,2001))
```

Now let's create and examine the design data to understand what has been done.

```
>Blackduck.ddl=make.design.data(Blackduck.process)
> Blackduck.ddl

$S
   group  age    time  Age Time year BirdAge
1  20000 0.25 2000.25 0.25 0.00 2000       0
2  20000  0.5  2000.5 0.50 0.25 2000       0
3  20000 0.75 2000.75 0.75 0.50 2000       0
4  20000    1    2001 1.00 0.75 2000       0
5  20000 1.25 2001.25 1.25 1.00 2000       0
6  20000  1.5  2001.5 1.50 1.25 2000       0
7  20000 1.75 2001.75 1.75 1.50 2000       0
8  20000    2    2002 2.00 1.75 2000       0
9  20010 0.25 2001.25 0.25 1.00 2001       0
10 20010  0.5  2001.5 0.50 1.25 2001       0
11 20010 0.75 2001.75 0.75 1.50 2001       0
12 20010    1    2002 1.00 1.75 2001       0
13 20010 1.25 2002.25 1.25 2.00 2001       0
14 20010  1.5  2002.5 1.50 2.25 2001       0
15 20010 1.75 2002.75 1.75 2.50 2001       0
16 20010    2    2003 2.00 2.75 2001       0
17 20001 1.25 2000.25 1.25 0.00 2000       1
18 20001  1.5  2000.5 1.50 0.25 2000       1
19 20001 1.75 2000.75 1.75 0.50 2000       1
```

```
20 20001    2    2001 2.00 0.75 2000       1
21 20001 2.25 2001.25 2.25 1.00 2000       1
22 20001  2.5  2001.5 2.50 1.25 2000       1
23 20001 2.75 2001.75 2.75 1.50 2000       1
24 20001    3    2002 3.00 1.75 2000       1
25 20011 1.25 2001.25 1.25 1.00 2001       1
26 20011  1.5  2001.5 1.50 1.25 2001       1
27 20011 1.75 2001.75 1.75 1.50 2001       1
28 20011    2    2002 2.00 1.75 2001       1
29 20011 2.25 2002.25 2.25 2.00 2001       1
30 20011  2.5  2002.5 2.50 2.25 2001       1
31 20011 2.75 2002.75 2.75 2.50 2001       1
32 20011    3    2003 3.00 2.75 2001       1
```

As you can see, each of the 4 groups (year-age) has 8 S parameters and it assigns the proper time and age values; although it is labeling based on the end of the interval unlike with φ. This allows easy modeling of age and time effects even though the cohorts overlap and start at different times and ages. While it is possible to do the same with **MARK**, the pre-defined models are not setup correctly and the necessary bookkeeping with the PIMS is even more difficult because the time is not the same for each column in the PIM. In **RMark**, we can easily create **`Age`** and **`Time`** models as follows:

```
> mark(Blackduck.process,Blackduck.ddl,model.parameters=list(S=list(formula=~Time)))

<...>

Real Parameter S
                                1         2         3         4         5
Group:year2000.BirdAge0 0.4662898 0.5396722 0.6113742 0.6785598 0.7390873
Group:year2001.BirdAge0 0.7390873 0.7917159 0.8360831 0.8725213 0.9018106
Group:year2000.BirdAge1 0.4662898 0.5396722 0.6113742 0.6785598 0.7390873
Group:year2001.BirdAge1 0.7390873 0.7917159 0.8360831 0.8725213 0.9018106
                                6         7         8
Group:year2000.BirdAge0 0.7917159 0.8360831 0.8725213
Group:year2001.BirdAge0 0.9249493 0.9429801 0.9568809
Group:year2000.BirdAge1 0.7917159 0.8360831 0.8725213
Group:year2001.BirdAge1 0.9249493 0.9429801 0.9568809

mark(Blackduck.process,Blackduck.ddl,model.parameters=list(S=list(formula=~Age)))

<...>

Real Parameter S
                                1         2         3         4         5
Group:year2000.BirdAge0 0.8116342 0.8024874 0.7930097 0.7832001 0.7730587
Group:year2001.BirdAge0 0.8116342 0.8024874 0.7930097 0.7832001 0.7730587
Group:year2000.BirdAge1 0.7730587 0.7625866 0.7517865 0.7406622 0.7292189
Group:year2001.BirdAge1 0.7730587 0.7625866 0.7517865 0.7406622 0.7292189
                                6         7         8
Group:year2000.BirdAge0 0.7625866 0.7517865 0.7406622
```

```
Group:year2001.BirdAge0 0.7625866 0.7517865 0.7406622
Group:year2000.BirdAge1 0.7174632 0.7054034 0.6930490
Group:year2001.BirdAge1 0.7174632 0.7054034 0.6930490

mark(Blackduck.process,Blackduck.ddl,model.parameters=list(S=list(formula=~Age+Time)))

<...>

Real Parameter S
                               1         2         3         4         5
Group:year2000.BirdAge0 0.6781671 0.6936336 0.7086762 0.7232748 0.7374130
Group:year2001.BirdAge0 0.9380706 0.9421129 0.9459066 0.9494650 0.9528010
Group:year2000.BirdAge1 0.2809199 0.2956494 0.3108173 0.3264028 0.3423816
Group:year2001.BirdAge1 0.7374130 0.7510774 0.7642580 0.7769480 0.7891434
                               6         7         8
Group:year2000.BirdAge0 0.7510774 0.7642580 0.7769480
Group:year2001.BirdAge0 0.9559270 0.9588550 0.9615962
Group:year2000.BirdAge1 0.3587260 0.3754053 0.3923856
Group:year2001.BirdAge1 0.8008429 0.8120479 0.8227619
```

C.21. Exporting to MARK interface

With the function **export.chdata** a processed **RMark** dataset can be exported to an **.inp** file for import into the **MARK** interface. You can use that to create a new analysis project in **MARK** and then use specific facilities in the **MARK** interface that are not in **RMark** (e.g., the **RELEASE** goodness of fit tests). In addition, with the function **export.model** you can export a specific model or a marklist of models from **RMark** for import into the **MARK** interface after creating. Exported models are simply renamed copies of the **marknnn.out**, **marknnn.res** and **marknnn.vcv** files. The files are named **MARKnnnY.tmp**, **MARKnnnX.tmp**, and **MARKnnnV.tmp** respectively. The models can be imported into the **MARK** interface by selecting **Output/Append** and selecting the output file (**MARKnnnY.tmp**). Only one model can be imported at a time.

As an example, the following will export the ubiquitous dipper data file and the models contained in **dipper.results** created in **example(dipper)**.

```
> data(dipper)
> dipper.processed=process.data(dipper,groups=("sex"))
> export.chdata(dipper.processed, filename="dipper")
> example(dipper)
> export.model(dipper.results)
```

C.22. Using R for further computation and graphics

One of the nice side benefits of **RMark** is that all of the power of **R** is available for further computation and plotting with the **MARK** output which is brought into the **mark** model object. We recommend exploring the various online sources of help with **R** graphics.

In addition to graphics, the entire **R** environment is available for further computation with the results. Some of this has already been done for you with functions like **covariate.predictions** (C.16) and **TransitionMatrix** (C.17) but there are always different computations that can follow once an analysis is completed. The most important feature about the choice of **R** for a statistical environment and the reason for its expansive growth is that it is open source. There is literally a worldwide collection of programmers who are continually developing and adding code to the **R** environment. **R** as an open source environment has at least 4 important consequences for the **R** user:

1. cutting edge analysis techniques are always being added to the environment
2. anything you probably need for computation has already been written so search before you write new code
3. all of the source code (including **RMark**) is available for you to examine so you can learn from other **R** programmers, modify it for your own needs and you can see how the code works
4. as a user it is up to you to make sure you are using it properly and to verify that the results are accurate or at least make sense.

This last point applies equally to commercial software but the advantage with open source software is that you can look at the code. We'll finish this paragraph with one last bit of soapbox commentary about science and software. The **R** environment is a useful model for the scientific community because it is open source and thus transparent and available to anyone willing to learn.

Enough of that! So let's explore an example that frequently appears on the **MARK** support forum which is the computation of a Delta method variance. Appendix B provides a thorough explanation of the underlying theory, and how a Delta method variance can be constructed 'by hand'. While it is obviously important to understand how the calculations are done, and the general limits of the method, once you have done one or two by hand there is little gain in learning. So once the learning is done why not automate what you need? Not surprisingly there is **R** code available to construct Delta method variances/covariances. As part of the **msm** package for analysis of multi-state Markov and hidden Markov processes, C. H. Jackson of the MRC Biostatistics Unit at Cambridge University provided a function called deltamethod for computation of Delta method variances of any function that can be differentiated with the **R** function **deriv** for symbolic differentiation of simple expressions. The code is quite simple because most of the work is in working out the derivatives and that is handled by the **deriv** function:

```
> deltamethod<- function (g, mean, cov, ses = TRUE)
{
    cov <- as.matrix(cov)
    n <- length(mean)
    if (!is.list(g))
        g <- list(g)
    if ((dim(cov)[1] != n) || (dim(cov)[2] != n))
        stop(paste("Covariances should be a ", n, " by ", n,
            " matrix"))
    syms <- paste("x", 1:n, sep = "")
    for (i in 1:n) assign(syms[i], mean[i])
    gdashmu <- t(sapply(g, function(form) {
        as.numeric(attr(eval(deriv(form, syms)), "gradient"))
    }))
```

```
    new.covar <- gdashmu %*% cov %*% t(gdashmu)
    if (ses) {
        new.se <- sqrt(diag(new.covar))
        new.se
    }
    else new.covar
}
```

If you install the **R** package **msm** (use **Packages/Install Packages**) from **CRAN**, then you can issue the command **library(msm)** to load the package and make the function **deltamethod** available in the **R** environment. In this example of using **deltamethod** we will compute the variances (or standard error, its square root) and covariances of φ and p from the **Phi(~1)p(~1)** model of the dipper data using the β's and their variances and covariances. This is not a particularly useful "further computation" but we chose it to show how **MARK** constructs these values and also to show that computation with the **deltamethod** function agrees with the **MARK** output. We start by fitting the model, displaying the summary with standard errors shown for the unique real parameters and extracting and displaying the β estimates:

```
> data(dipper)
> mymodel=mark(dipper,brief=TRUE)

Model: Phi(~1)p(~1)  npar= 2  lnl =  666.83766 AICc = 670.86603

> summary(mymodel,se=TRUE,showall=FALSE)

Output summary for CJS model
Name : Phi(~1)p(~1)

Npar :  2
-2lnL:  666.8377
AICc :  670.866

Beta
                 estimate        se       lcl       ucl
Phi:(Intercept) 0.2421484 0.1020127 0.0422035 0.4420933
p:(Intercept)   2.2262658 0.3251093 1.5890516 2.8634801


Real Parameters
                 estimate        se       lcl       ucl fixed
Phi g1 c1 a0 t1 0.5602430 0.0251330 0.5105493 0.6087577
p g1 c1 a1 t2   0.9025835 0.0285857 0.8304826 0.9460113

> betas=summary(mymodel)$beta
> betas
                 estimate        se       lcl       ucl
Phi:(Intercept) 0.2421484 0.1020127 0.0422035 0.4420933
p:(Intercept)   2.2262658 0.3251093 1.5890516 2.8634801
```

We can see the confidence intervals for β's are simple normal 95% confidence intervals:

```
> beta.lcl=betas$estimate-1.96*betas$se
> beta.lcl
[1] 0.04220351 1.58905157
> beta.ucl=betas$estimate+1.96*betas$se
> beta.ucl
[1] 0.4420933 2.8634800
```

Now let's compute the confidence intervals for the real parameters which are the inverse **logit** (in general inverse of the chosen link function) of the lower and upper limits on the β's.

```
> exp(beta.lcl)/(1+exp(beta.lcl))
   [1] 0.5105493 0.8304826
> exp(beta.ucl)/(1+exp(beta.ucl))
   [1] 0.6087577 0.9460113
```

We can individually compute the standard errors for the real parameters with calls to **deltamethod** using the inverse **logit** function where **x1** refers to **beta**:

```
> deltamethod(~exp(x1)/(1+exp(x1)),mean=betas$estimate[1],cov=betas$se[1]^2)
   [1] 0.02513295
> deltamethod(~exp(x1)/(1+exp(x1)),mean=betas$estimate[2],cov=betas$se[2]^2)
   [1] 0.02858573
```

We can get the same results with a single call to **deltamethod** by using a list of functions and the variance-covariance matrix for **beta** which is in **results$beta.vcv**:

```
> deltamethod(list(~exp(x1)/(1+exp(x1)),~exp(x2)/(1+exp(x2))),
             mean=betas$estimate,mymodel$results$beta.vcv)
    [1] 0.02513295 0.02858573
```

or we can get the variance-covariance matrix of the real parameters by setting **ses=FALSE**:

```
> deltamethod(list(~exp(x1)/(1+exp(x1)),~exp(x2)/(1+exp(x2))),
      mean=betas$estimate,mymodel$results$beta.vcv,ses=FALSE)
               [,1]          [,2]
 [1,]  0.0006316653 -0.0001842868
 [2,] -0.0001842868  0.0008171439
```

For closed population modeling, the **R** package **WiSP** (Wildlife Simulation Package, available from **http://www.ruwpa.st-and.ac.uk/estimating.abundance/WiSP/index.html**) can be used to simulate populations and sampling designs. Data simulated by **WiSP** can be converted into a form usable by **RMark** using a conversion function **transform.to.rmark()** found in the **WiSP** package. This enables the examination of estimator performance prior to the conduct of field experiments.

C.23. Problems and errors

The **RMark** code includes some error traps but there are a number of errors that can occur if the models or data are not setup properly. In no particular order, we give some errors that can occur, an explanation,

and some possible fixes. We do not discuss **R** syntax errors which can occur easily if improper syntax is used. We suggest that you use an editor like **Tinn-R** (available from **https://sourceforge.net/projects/tinn-r**) that provides **R** syntax checking to develop scripts.

1. The following error message or one like it that occurs when **mark.exe** is running or afterwards, occurs when something is amiss with the data, model setup for **MARK** or you interrupted the job:

```
Error in if (x4 > x2) { : argument is of length zero

********Following model failed to run : [name of model]**************
S.dot.p.dot.Psi.dot
Error in extract.mark.output(out, model, adjust) :
MARK did not run properly. If error message was not shown, re-run
 MARK with invisible=FALSE
```

   Solution: Look at the most current input and output files in the directory to see if you can discern what happened. Error messages and the output will often move across the screen too quickly to read but you can always look at them with a text editor to discover the reason for the problem. The obtuse error above occurs because the output file is incomplete and the function **extract.mark.output** cannot find relevant fields in the output file.

2. The following error message occurs when a variable used in a formula cannot be found in the design data or as an individual covariate:

```
Variable marked.as.adult used in formula is not defined in data
Error in make.mark.model(data.proc, title = title, covariates = covariates,
```

   Solution: Check your the spelling of your variable name. Remember that **R** is case specific so check capitalization. If you added design data, make sure that you added the data to the design data for parameter that is generating the error for this variable. If it is a time-varying covariate make sure that the times match the prefixes of the covariate names.

3. If you get an error like the following, you have created an incomplete factor variable in the design data.

```
Error in make.mark.model(data.proc, title = title, covariates = covariates,  :

Problem with design data. It appears that there are NA values
in one or more variables in design data for p
Make sure any binned factor completely spans range of data
```

   Solution: Examine the design data identified in the error message and redefine the factor variable.

4. If you get either of these error messages, there is a problem with the individual covariate that you have specified in the formula.

```
The following individual covariates are not allowed because
they are factor variables:
The following individual covariates are not allowed because
they contain NA:
```

Solution: Use **summary(data)** where data is the dataframe containing your data. Examine the variable it names to see where it contains NA or if it is a factor variable. A factor variable cannot be used as an individual covariate. It is best to use factor variables in the group structure definition. If you want to use a factor variable as an individual covariate you need to create numeric dummy variables (0/1). See section C.16.

5. The following error occurs when you attempt to fix real parameters and the number of values does not match the number of indices.

```
Lengths of indices and values do not match for fixed parameters for p
```

Solution: Refer to section C.11 to review how real parameters can be fixed at specific values.

6. The following error occurs when you specify a vector of initial values but the the number of values does not match the number of β's (number of columns in the design matrix).

```
Length of initial vector doesn't match design matrix
```

Solution: Use another existing model to specify the initial values. or use **model.matrix** to compute the number of parameters will be fit. If the model has already run, extract the β estimates into a vector to verify the count or edit it and use as the initial values.

7. The following error occurs when you specify a formula that manages to create a design matrix which has all zeros for one or more real parameters (rows). This will most likely occur with the incorrect specification of interactions and the intercept is removed.

```
One or more formulae are invalid because the design matrix has all
zero rows for the following non-fixed parameters
```

Solution: Use **model.matrix** with the formula and design data and review the way you are constructing the formula.

8. The following error will occur if you pass the wrong data argument to **make.design.data**

```
Error in if (model == "CJS") par.list = c("Phi", "p") :
  argument is of length zero
```

Solution: Use the processed dataframe and not the original dataframe in the call the **make.design.data**.

It is reasonable to check an **RMark** model by creating it with the **MARK** interface and compare the results. If you do that, recognize that both interfaces use the same **mark.exe** to fit the model to the data. Thus, if there is a difference then it results from a difference in either the data or the model structure. Presumably, you are using the same data but it doesn't hurt to check the output from each model to see that it used the same data. A difference in the model is the most likely reason for any difference. If the deviance and the real parameters match but the β's are different, that is not a real concern because the same model can be fit with different beta structures. The differences in the β's can occur if different link functions are chosen or a different structure for the design matrix was used. However, if the deviances

or real parameters are different then it should be investigated further. A difference in the link function can create difference in the real parameters if there is a problem with convergence of one of the models. However, the most likely difference is an error in the PIM structure or design matrix. Only a tedious search of the PIMS and design matrices will identify the problem. While it is always possible that there is an error in the **RMark** code, numerous examples have been tested in both pieces of software to check agreement.

C.24. A brief R primer

There are a number of very good books and tutorials for learning **R**. See the **R** home page at http://www.r-project.org/ and browse the links under Documentation on the left panel. If you have questions, refer to the FAQ or use the Search utility. They are available from **R** Homepage or from within **R** under the Help menu. There is a phenomenal amount of material on the web that can help you get started with R. If you have problems and cannot find the answer with the materials on the web, a very active user group can be found on the R-help list server (see Mailing Lists on the home page). If you subscribe and post messages, please read the posting guide! The search utilities will search the **R**-help list server. There is a good chance your question has already been answered, so please read the FAQ and search before you post a question.

We have no intention of producing a full **R** tutorial here but we will provide some very beginner concepts and some others that are particularly relevant to using **RMark**. You can start **R** with the **R** icon or by double-clicking a **.Rdata** file which is an **R** workspace where everything is stored by **R**. If you start **R** with the icon, it will use the default **.Rdata** workspace located where you installed **R** (typically **c:/Program Files/R/Rvvvvv**, where **vvvvv** represents the version). You will most likely want to have more than one **.Rdata** workspace and there are many ways to create them, but the simplest is to use Windows to copy the default workspace and to paste it in whatever directory you choose. When you then double-click that particular **.Rdata** file, it will open it with **R**.

To avoid manually entering the command each time you initiate **R**, you can edit and enter the **library(RMark)** command into the file named "**RProfile.site**" with any text editor. It is located in the directory

```
C:\Program Files\R\R-v.v.v\etc\
```

where **v.v.v** represents the **R** version. If you add the library(RMark) command to rprofile.site, the RMark package will be loaded anytime you start R.The **RProfile.site** file is also a good place to make generic customizations to **R**. For example, adding the command **options(chmhelp=TRUE)** will mean that the help command will use the compiled help tool for windows. Or you can use options **(htmlhelp=TRUE)** to use the non-compiled html help. Either of those is better than using the default which does not allow hyperlinks. If you set up either help option you can enter "**?mark**" to see all the help categories for **RMark**. If you chose to use **htmlhelp**, click on index to see the complete list. The "**rprofile.site**" file is also a good place to change options like the default editor etc. See help for "options" and "Startup" from within **R**.

To avoid making changes to **RProfile.site** each time you update **R**, you can use the Windows ControlPanel and select System/Advanced/EnvironmentVariables to create an environment variable named **R_PROFILE**. For example, you can create a subdirectory,

```
C:\Program Files\R\RProfile\
```

and then copy your edited **Rprofile.site** to that subdirectory. Then define the environment variable

R_PROFILE with the value

```
C:\Program Files\R\RProfile\RProfile.site
```

and it will always be used for each **R** session even if you update **R**.

You quit **R** with the command **q()** and it will ask whether you want to save the workspace image. If you select **No**, then any **R** objects you created/deleted/changed during the session will not be saved. You can save the workspace during the session with the File/Save Workspace or using the disk icon on the toolbar. This is not a bad idea to avoid losing work.

R is case-sensitive. **Q** and **q** are not the same. Some functions (e.g., **rowSums**) will even mix cases in the function name so be aware. Object names can have mixed cases, periods and underscore to improve readability (e.g., **my.list**, **squirrel_results**, **DipperModels**).

The symbol # is used for comments and anything to the right of the # is ignored in a line. You'll see them in examples below and in the help files for **RMark** and other help files.

The parentheses after **q** are necessary. Try typing **q** without the parentheses and what you'll get is a listing of the function "**q**". However when you type **q()**, it executes the function "**q**", and "**()**" represents the arguments for the function which happens to be empty in this case.

Almost everything you do in **R** will be executing functions that accept arguments and return a value. The functions and the values returned by executing functions can all be stored as named objects in the workspace. Assignment of function values to an object is done with the assignment operator which can be either **<-** or = (or **<<-** for global assignment within functions). Typing **x=1+1** or **x <- 1+1**, assigns the numeric result 2 to the object named *x*. Typing **x=mean(1:50)**, assigns the mean of the sequence of numbers from 1 to 50 to *x*. The definition of a function is an assignment of **R** code to an object with a specific function name. You can see a listing of the names of the objects in the workspace by typing **ls()**.

If you execute a function and do not assign the result (if any) to an object, a default **print** function for that object will display it on the screen or to a file (if you use the **sink** function) and nothing becomes of it. The function **ls()** is a good example. As it says in the help file for **ls**, the function returns a vector of character strings giving the names of the objects in the specified environment. That vector of character strings can be assigned to an object but if you simply type **ls()**, the character string is printed (displayed) on the screen.

Within **R** you can get help for any function by simply typing **?** followed by the function name. For example, **?ls**. If you don't know the name of the function but have some keywords to describe it, use the function **help.search("my key words")** or browse through the help manuals which can be accessed from the Help menu. A reference card of commonly used **R** functions can also be useful - see

http://cran.r-project.org/doc/contrib/Short-refcard.pdf

You can access the full help file for **RMark** by opening the file **RMark.chm** (compiled help file) contained in the install directory **c:/Program Files/R/Rvvvvv/library/RMark/chm** (where **vvvvv** represents the **R** version number). If you only want to know the arguments of a function, you can use the function **args(function)** to list the argument names and default values of the function. For example, type **args(ls)** to see that the **ls** function does have arguments but the default values are typically used, so you only need to type **ls()**.

Values for function arguments can be assigned by their order in the function call and by specifying argument=value. The advantage of the latter format is that it is not order-specific. Both formats can be used in the same function call and it is a typical choice to specify the first few common arguments by

order and then specifying less often used arguments by name. As an example, we will use the function **rnorm** which generates random values from a normal distribution. If you type **args(rnorm)**, you'll see the following: **function (n, mean = 0, sd = 1)**. This means that the function has 3 arguments named **n**, **mean** and **sd**. The latter 2 have default values of 0 and 1, so if you don't specify values for those arguments they will be assigned 0 and 1 respectively. The argument "**n**" is the number of random values to be generated and its value must be given because it has no default. By typing **rnorm(100)**, you will generate $n = 100$ random values from a standard normal distribution with mean 0 and standard deviation 1. If you don't assign them to an object they will simply be displayed on the screen. If you wanted an **sd** of 5, you could do that by either of the following calls: **rnorm(100,,5)** or **rnorm(100,sd=5)**. The first form assigns the argument values solely by position in the argument list and the second uses both formats with n being assigned 100 because of its first position and sd is assigned 5 with a named value. Naming arguments does make the code more readable and you could choose to specify all the arguments by name with **rnorm(n=100, mean=0, sd=5)** and that is perfectly suitable. If you forget to assign a value to an object which does not have a default, an error will be issued. For example, you may see something like the following:

```
> rnorm(mean=2)
Error in rnorm(mean = 2) : argument "n" is missing, with no default
```

You can write your own functions that are stored in the workspace, but most functions you will use are in base **R** packages or other contributed packages that can be optionally installed with **R**. The multitude of contributed packages on CRAN (Comprehensive **R** Archive network) can be found from the **R** home page. While most contributed packages are on CRAN, there are other supported packages such as **RMark** that can be found on the web. Packages must be installed and then loaded with the **library()** function as described above for **RMark**. Once you have installed **RMark** and used the library(**RMark**) function you can access the help and use the functions. Remember that **R** is case-sensitive so **RMark** is not the same as **Rmark**.

There are several different kinds of data structures in **R**. The most basic is a vector and while most objects in **R** are generalizations of vectors, here we are referring to a vector as an ordered collection of items of the same type. For example, **c(4,2,1)** is a numeric vector with the first item being 4, the second 2 and the third being 1. As you might expect, **c()** is a function - the concatenate function that puts together items of the same type or coerces them to the same type. Numeric vectors can be created with sequences such as **1:5** which is the sequence from 1 to 5 and by various functions (e.g., **seq**, **rep**). Vectors can also contain character or logical values. For example, **c("apple","orange","grape")** is a vector of character strings of fruit names. Logical vectors are typically created by comparison operators such as equality (==), not equal (!=), less than (<), greater than (>), not greater than (<=), and not less than (>=). Both of the following commands create a logical vector of length 5 with different values:

```
> 1:5 ==2
[1] FALSE  TRUE FALSE FALSE FALSE
> 1:5 >2
[1] FALSE FALSE  TRUE  TRUE  TRUE
Logical values (vector of length 1) can also be created with the %in% operator:
> "orange" %in% c("apple","orange","grape")
[1] TRUE
> "pineapple" %in% c("apple","orange","grape")
[1] FALSE
```

Other data structures include matrices, arrays, dataframes (tables) and lists. Matrices are rectangular structures with each element restricted to be the same type (e.g., numeric, character, logical) and arrays are generalizations of matrices to higher dimensions. While matrices are used in **RMark** (e.g., design matrix), the primary data structures are lists and dataframes. You need to know how to construct and manipulate both types of data structures to be able to run models other than the most rudimentary ones.

Lists are the most predominant data structure in **RMark** because they are used to specify argument values and most functions return lists. Dataframes are special cases of lists, so we'll start by describing lists. A list is a collection of data structures that are not restricted to be the same type/structure. A list can contain numeric vectors, character vectors, a matrix, other lists and so on and so forth. It is a truly generic structure that lets you paste together different kinds of data structures. A list is often the value returned from a function because a function can only return a single object and often it is the only way to return many different types of values by pasting them into a single list. Likewise, lists can be used to paste together different types of data (e.g, character, numeric, logical) that are related to be passed as value for a function argument.

Everyone is familiar with grocery and "to do" lists so we'll use them as examples. To create a very strange grocery list in **R** called groceries you would use the **`list()`** function:

```
> groceries=list(fruits=c("oranges","apples","grapes"),
            meat=c("steak", "chicken"), milk=c(1,.5) )
```

If you wanted to see the contents of groceries it would look as follows:

```
> groceries
$fruits
[1] "oranges" "apples"  "grapes"

$meat
[1] "steak"   "chicken"

$milk
[1] 1.0 0.5
```

The groceries list has length 3 as you can ascertain with the length function:

```
> length(groceries)
[1] 3
```

It contains 3 vectors with the first being character vectors named "**`fruits`**" and "**`meat`**" and the third is a numeric vector containing the size in gallons and named "**`milk`**". The first vector contains 3 elements, and the second and third vectors each contain 2 elements. You can extract the **`meat`** vector in several ways.

```
> groceries$meat
[1] "steak"   "chicken"
>
> groceries[[2]]
[1] "steak"   "chicken"
```

```
>
> groceries[["meat"]]
[1] "steak"   "chicken"
```

The double-square brackets are used to extract a list element by number or name. What is returned is the contents of the list element which is a character vector in this case. Now notice what happens when you use single brackets instead:

```
> groceries[2]
$meat
[1] "steak"   "chicken"

> groceries["meat"]
$meat
[1] "steak"   "chicken"
```

Single brackets provide a subset of the list and the result is a list and not just the contents of the list. The difference between the single and double brackets is shown below with the is.list function. With single brackets the result is a list and with double brackets it is not a list.

```
> is.list(groceries["meat"])
[1] TRUE
> is.list(groceries[["meat"]])
[1] FALSE
```

In most cases with **RMark** you will use the [[]] to extract the contents of a single list element. On some occasions, you would like to extract several list elements and this is done with the single brackets:

```
> groceries[c("meat","milk")]
$meat
[1] "steak"   "chicken"

$milk
[1] 1.0 0.5

> groceries[2:3]
$meat [
1] "steak"   "chicken"

$milk
[1] 1.0 0.5
```

In both cases above, a list with 2 elements is returned. If you try to extract more than one list element with [[]], an error will result:

```
> groceries[[2:3]]
Error in groceries[[2:3]] : subscript out of bounds

> groceries[[c("meat","milk")]]
Error in groceries[[c("meat", "milk")]] : subscript out of bounds
```

In **RMark**, you'll most often extract specific list elements either by name (**groceries$meat**) or by position (**groceries[[2]]**).

Lists can contain lists as elements and while this can look rather bizarre at first, it is extremely handy. Assume that we had a "to do" list as follows:

```
todo=list(for.wife=c("grocery","cut grass","dry cleaner"),
for.me=c("watch tv","drink beer", "nap"))
```

If we concatenate the lists it merges them into a single list with all the different elements:

```
> c(todo,groceries)
$for.wife
[1] "grocery"     "cut grass"   "dry cleaner"

$for.me
[1] "watch tv"   "drink beer" "nap"

$fruits
[1] "oranges" "apples"  "grapes"

$meat
[1] "steak"   "chicken"

$milk
[1] 1.0 0.5
```

Alternatively, we can also create a list of lists as follows:

```
> mylists=list(todo=todo,groceries=groceries)
> mylists
$todo
$todo$for.wife
[1] "grocery"     "cut grass"   "dry cleaner"

$todo$for.me
[1] "watch tv"   "drink beer" "nap"

$groceries
$groceries$fruits
[1] "oranges" "apples"  "grapes"

$groceries$meat
[1] "steak"   "chicken"

$groceries$milk
[1] 1.0 0.5
```

I can extract each sub-list as described previously:

```
> mylists$todo
$for.wife
[1] "grocery"     "cut grass"   "dry cleaner"

$for.me
[1] "watch tv"   "drink beer" "nap"

> mylists$groceries
$fruits
[1] "oranges" "apples"  "grapes"

$meat
[1] "steak"   "chicken"

$milk
[1] 1.0 0.5
```

Lists of lists are used to provide a generic structure in **RMark** to accommodate varying parameters within the different **MARK** models. Likewise, design data for the parameters is represented as a list of dataframes.

That brings us to the final data structure we'll discuss. A dataframe is a specialized list in which each element is a named vector and each vector is of the same length but not necessarily of the same type. A dataframe is rectangular like a matrix. In a dataframe, all the values in a column (the list element vectors) are of the same type but one column can be numeric, the next column could be character and another could be logical. It is easiest to conceptualize dataframes as similar to tables like in ACCESS or any other database package.

We can show the link between lists and dataframes by using the **as.data.frame** function to convert the todo list to a dataframe.

```
> todo.data=as.data.frame(todo)
> todo.data
     for.wife     for.me
1     grocery   watch tv 2   cut grass drink beer 3 dry cleaner nap
```

The columns of the dataframe **todo.data** are the 2 vectors contained in the **todo** list. This worked because each vector was of the same length. If we did the same conversion with the **groceries** list, an error occurs because the vectors are of unequal length:

```
> as.data.frame(groceries)
Error in data.frame(fruits = c("oranges", "apples", "grapes"), meat
= c("steak",  :
        arguments imply differing number of rows: 3, 2
```

Notice that in the conversion from list to dataframe the quotation marks around the character strings vanished. Dataframes were designed for analysis and character strings aren't typically very useful for analysis but character strings can be used to represent the names of factor variable levels. By default, the character strings were coerced into a factor variable:

```
> todo.data$for.me
[1] watch tv   drink beer nap Levels: drink beer nap watch tv
> is.factor(todo.data$for.me)
[1] TRUE
```

The columns (variables) of **todo.data** are factor variables and the names of the levels for the factor variable **for.me** are "**drink beer**", "**nap**" and "**watch tv**". Note that the levels are alphabetized by default and the numeric values of **for.me** are determined by the order of the levels:

```
 > as.numeric(todo.data$for.me)
[1] 3 1 2
```

In **RMark**, dataframes are used for the capture history and related covariate data for animals. They are also used for design data which describes the model structure.

APPENDIX D

Variance components and random effects models in MARK . . .

Kenneth P. Burnham, *USGS Colorado Cooperative Fish & Wildlife Research Unit*

The objectives of this appendix are

- to introduce to biologists the concept and nature of what are called (alternative names for the same essential idea) 'variance components', 'random effects', 'random coefficient models', or 'empirical Bayes estimates'
- present the basic theory and methodology for fitting simple random effects models, including shrinkage estimators, to capture-recapture data (i.e., Cormack-Jolly-Seber and band or tag recovery models)
- extend AIC to simple random effects models embedded into the otherwise fixed-effects capture-recapture likelihood.
- develop some proficiency in executing a variance components analysis and fitting random effects model in program **MARK**

Much of the conceptual material presented in this appendix is derived from a paper authored by Kenneth Burnham and Gary White (2002) – for convenience, we will refer to this paper as 'B&W'. It is assumed that the reader already has a basic knowledge of some standard encounter-mark-reencounter models as described in detail in this book (e.g., dead recovery and live recapture models – referred to here generically as capture-recapture).

We introduce the subject of – and some of the motivation for – this appendix by example. In the following we consider two relatively common scenarios (out of a much larger set of possibilities) where a 'different analytical approach' might be helpful.

scenario 1 – parameters as random samples

Consider a Cormack-Jolly-Seber (CJS) time-specific model $\{S_t p_t\}$ wherein survival (S) and capture probabilities (p) are allowed to be time varying for $(k + 2)$ capture occasions, equally spaced in time. If $k \geq 20$ we are adding many survival parameters into our model as if they were unrelated; however, more parsimonious models are often needed. Consider a reduced parameter model – at the extreme, we have the model $\{S_\cdot p_t\}$ wherein $S_1 = S_2 = \cdots = S_k = S_\cdot$. However, this model may not fit well even if

© Cooch & White (2014)

the general (time-dependent) CJS model fits well and there is no evidence of any explainable structural time variation, such as a linear time trend, in this set of survival rates, or variation as a function of an environmental covariate. Instead, there may be unstructured time variation in the S_i that is not easily modeled by any simple smooth parametric form, yet which cannot be wisely ignored. In this case it is both realistic and desirable to conceptualize the actual unknown S_i as varying, over these equal-length time intervals, about a conceptual population mean $E(S) = \mu$, with some population variation, σ^2 (Fig. E.1).

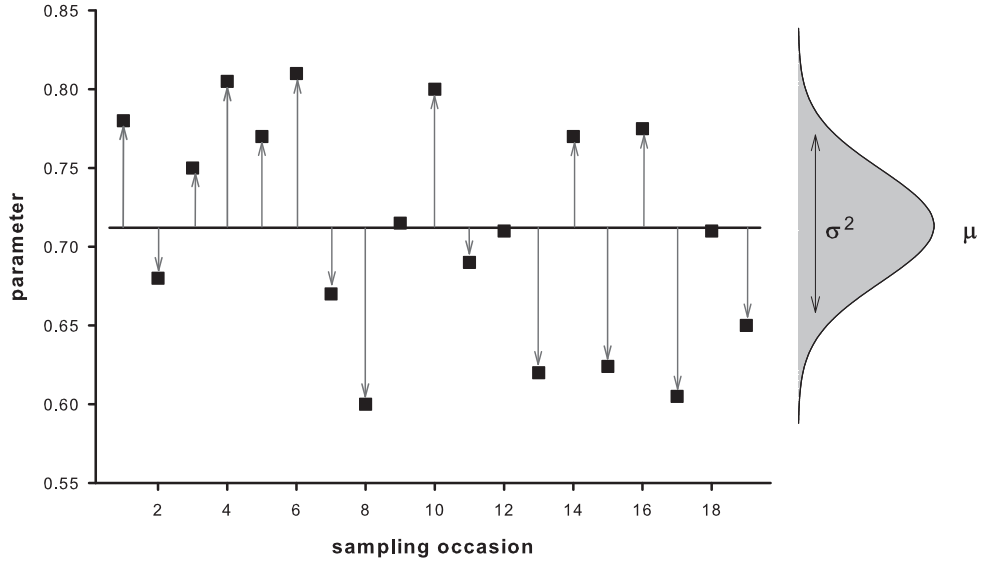


Figure D.1: *Schematic representation of variation in occasion-specific parameters θ_i, as if the parameters were drawn randomly from some underlying distribution with mean μ and variance σ^2.*

Here, by population, we will mean a conceptual statistical distribution of survival probabilities, such that the S_i may be considered as a sample from this distribution. Hence, we proceed as if S_i are a *random* sample from a distribution with mean μ and variance σ^2. Doing so can lead to improved inferences on the S_i regardless of the truth of this conceptualization if the S_i do in fact vary in what seems like a random, or exchangeable, manner. The parameter σ^2 is now the conventional measure of the unstructured variation in the S_i, and we can usefully summarize $S_1 \ldots S_k$ by two parameters: μ and σ^2. The complication is that we do not know the S_i; we have only estimates $\hat{S}_i$, subject to non-ignorable sampling variances and covariances, from a capture-recapture model wherein we traditionally consider the S_i as fixed, unrelated parameters. We would like to estimate μ and σ^2, and adjust our estimates to account for the different contributions to the overall variation in our estimates due to sampling, and the environment. For this, we consider a random effects model.

scenario 2 – separating sampling + environmental (process) variation

Precise and unbiased estimation of parameter uncertainly (say, the SE of the parameter estimate) is critical to analysis of stochastic demographic models. Consider for example, the estimation of the risk of extinction. It is well known (and entirely intuitive) that any simply stochastic process (say, growth of an age- or size-structured population through time) is more likely to go extinct the more variable a

particular 'vital rate' is (say, survival or fertility). Thus, if an estimate of the variance of a parameter is biased high, then this tends to bias high the probability of extinction. We wish to use only estimates of environmental (or process) variation alone, excluding sampling variation, since it is only the magnitude of the former that we want to include in our viability models (White 2000).

Precise estimation of process variation is also critical for analysis of the relationship of the variation of a particular demographic parameter to the projected growth of a population. The process variation in projected growth, λ, is a function of the process variance of a particular demographic parameter. To first order, and assuming no covariances between the a_{ij} elements, this can be expressed as

$$\widehat{\mathrm{var}}(\lambda) \approx \sum_{ij} \left(\frac{\partial \lambda}{\partial a_{ij}} \right) \widehat{\mathrm{var}}(a_{ij})$$

From this expression, we anticipate that natural selection will select against high process variation in a parameter that λ is most 'sensitive' to (i.e., for which $\partial\lambda/\partial a_{ij}$ is greatest) (Pfister 1998; Schmutz 2009). Thus, robust estimation of the process variance of fitness components is critical for life history analysis.

In this appendix, we will consider estimation of variance components, and fitting of random effects models, using program **MARK**. We begin with the development of some of the underlying theory, followed by illustration of the 'mechanics' of using program **MARK**, by means of a series of 'worked examples'.

D.1. variance components - some basic background theory

The basic idea is relatively simple. We imagine that the S_i are distributed randomly about $E(S) = \mu$ (Fig. E.1). The variation in S_i is σ^2, as if $S_1, \ldots S_k$ are a sample from a population. It is not required that the sampling be random – merely that the $S_1, \ldots, S_k$ are exchangeable (or, more formally, that the conceptual residuals $(\mu - S_i)$ should appear like an iid sample, with no remaining structural information). There are no required distributional assumptions, such as normality.

If we knew the S_i then it follows that

$$\hat{E}(S) = \bar{S} \qquad \hat{\sigma}^2 = \frac{\sum^k (S_i - \bar{S})^2}{k-1}$$

Of course, except in a computer simulation, we rarely if ever know the S_i. What we might have are ML estimates $\hat{S}_i$, and estimates of conditional variation $\widehat{\mathrm{var}}(\hat{S}_i \mid S_i)$.

We can express our estimate of $\hat{S}_i$ in standard linear form as the sum of the mean μ, the deviation of the S_i from the mean, δ_i, and the error term ϵ_i

$$\hat{S}_i = \mu + \delta_i + \epsilon_i$$

where $\delta_i = (S_i - \mu)$ and $\epsilon_i = (\hat{S}_i - S_i)$. Substituting into our expression for $\hat{S}_i$,

$$\begin{aligned} \hat{S}_i &= \mu + \quad \delta_i \quad + \quad \epsilon_i \\ &= \mu + \underbrace{(S_i - \mu)}_{\uparrow \sigma^2 \text{ (process variance)}} + \underbrace{(\hat{S}_i - S_i)}_{\uparrow \mathrm{var}(\hat{S}_i \mid S_i) \text{ (sampling variance)}} \end{aligned}$$

Here (and hereafter) we distinguish between '*process*' (or, environmental) variation, σ^2, and '*sampling*' variation, $\text{var}(\hat{S}_i \mid S_i)$. We refer to the sum of process and sampling variation as total variation, σ^2_{total}.

$$\begin{aligned}\text{total variation} = \sigma^2_{total} &= \text{process variation} + \text{sampling variation} \\ &= \sigma^2 + \text{var}(\hat{S}_i \mid S_i)\end{aligned}$$

It is important to note that sampling variation $\text{var}(\hat{S}_i \mid S_i)$ depends on the sample size of animals captured, whereas process variance σ^2 does not. It is also important to note that if there is sampling covariation, then this should be included in our expression for total variance, σ^2_{total}

$$\sigma^2_{total} = \sigma^2 + \left[E\Big(\text{var}(\hat{S}_i \mid S_i)\Big) + E\Big(\text{cov}(\hat{S}_i, \hat{S}_j \mid S_i, S_j)\Big)\right]$$

For fully time-dependent models, the sampling covariances of S_i and S_j are often very small for many of the data types we work with in **MARK**, especially relative to process and sampling variance, and the covariance term can often be ignored. We will do so now, for purposes of simplifying the presentation somewhat, but will return to the issue of sampling covariances later on.

If we assume for the moment that all the sampling variances are equal, then the estimate of the overall mean survival is just the mean of the k estimates:

$$\bar{\hat{S}} = \frac{\sum^k \hat{S}_i}{k}$$

with the theoretical variance being the sum of process and sampling variance divided by k:

$$\widehat{\text{var}}(\bar{\hat{S}}) = \frac{\sigma^2 + E\Big[\text{var}(\hat{S}_i \mid S_i)\Big]}{k}$$

Our interest generally lies in estimation of the process variation. By algebra, we see that process variance can be estimated by, in effect, subtracting the sampling variation from the total variation.

$$\begin{aligned}\sigma^2_{total} &= \text{process variation} + \text{sampling variation} \\ &= \sigma^2 + \widehat{\text{var}}(\hat{S}_i \mid S_i) \\ \therefore \sigma^2 &= \sigma^2_{total} - \widehat{\text{var}}(\hat{S}_i \mid S_i)\end{aligned}$$

Hence, we need an estimate for σ^2_{total} and $\text{var}(\hat{S}_i \mid S_i)$.

If we assume that $S_1, \ldots, S_k$ are a random sample, with $\bar{S} = \hat{E}(S)$, and population variance σ^2, then from general least-squares theory

$$\bar{\hat{S}} = \frac{\sum^k w_i \hat{S}_i}{\sum^k w_i}$$

where

$$w_i = \frac{1}{\sigma^2 + \widehat{\text{var}}(\hat{S}_i \mid S_i)}$$

Given w_i, the theoretical variance of $\hat{\bar{S}}$ is

$$\mathrm{var}(\hat{\bar{S}}) = \frac{1}{\sum^k w_i}$$

However, although $\mathrm{var}(\hat{S}_i \mid S_i)$ is estimable, we would still need to know σ^2.

An alternative approach which leads to an empirical (data-based) estimator is

$$\widehat{\mathrm{var}}(\hat{\bar{S}}) = \frac{\sum^k w_i(\hat{S}_i - \hat{\bar{S}})^2}{(\sum^k w_i)(k-1)}$$

We note that if there is no process variation ($\sigma^2 = 0$), then the above reduces to the familiar case of k replicates.

More generally, if we assume that the weights, w_i are equal (or nearly so), then we can re-write the empirical estimator as

$$\widehat{\mathrm{var}}(\hat{\bar{S}}) = \frac{\sum^k (\hat{S}_i - \hat{\bar{S}})^2}{(k-1)}, \qquad \text{if } w_i = w, \forall_i$$

where

$$\hat{\bar{S}} = \frac{\sum^k \hat{S}_i}{k}$$

The assumption that the weights, w_i are equal is generally reasonable if (*i*) the $\mathrm{var}(\hat{S}_i \mid S_i)$ are all nearly equal, or if (*ii*) they are all small, relative to σ^2. In theory, when the S_i vary, then the $\mathrm{var}(\hat{S}_i \mid S_i)$ will also vary. In contrast, with low sampling effort, such that $\mathrm{var}(\hat{S}_i \mid S_i)$ is much larger than process variance σ^2, it might be sufficient to use the approximation

$$w_i = \frac{1}{\mathrm{var}(\hat{S}_i \mid S_i)}$$

In this case, only relative weights w_i would be needed (since an estimate of σ^2 is not needed).

Now, assume for the moment we are interested in estimating the process variation around the mean $\bar{S}$. If we also assume that there is no sampling covariance, and that $w_i = w$, and $\sum^k w_i = 1$, then we estimate the total variance as

$$\sigma^2_{total} = \widehat{\mathrm{var}}(\hat{\bar{S}}) = \frac{\sum^k (\hat{S}_i - \bar{\hat{S}}_i)^2}{(k-1)}$$

and the sampling variance as the mean of the estimated sample variances

$$E\left[\widehat{\mathrm{var}}(\hat{S}_i \mid S_i)\right] = \frac{\sum^k \mathrm{var}(\hat{S}_i \mid S_i)}{k}$$

Hence, our estimate of process variance would be

$$\hat{\sigma}^2 = \frac{\sum^{k-1} (\hat{S}_i - \bar{\hat{S}}_i)^2}{k-1} - \frac{\sum^k \mathrm{var}(\hat{S}_i \mid S_i)}{k}$$

However, this estimator (which is essentially the estimator described by Gould & Nichols, 1998*) is not entirely correct (or efficient). It was derived under the strong assumption that the sampling variances are all equal (i.e., that $SE(\hat{S}_i)$ are all identical). In practice, this is usually not the case, and thus we refer to the preceding as a 'naive' estimator of process variance.

Instead, we need to weight them to obtain an unbiased estimate of σ^2. As noted earlier, general least-squares theory suggests using a weight w_i

$$w_i = \frac{1}{\sigma^2 + \mathrm{var}(\hat{S}_i \mid S_i)}$$

Hence, the estimator of the *weighted* mean survival is

$$\bar{\hat{S}} = \frac{\sum^k w_i \hat{S}_i}{\sum^k w_i}$$

with theoretical variance

$$\mathrm{var}(\bar{\hat{S}}) = \frac{1}{\sum^k w_i}$$

and empirical variance estimator

$$\widehat{\mathrm{var}}(\bar{\hat{S}}) = \frac{\sum^k w_i (\hat{S}_i - \bar{\hat{S}})^2}{\left[\sum^k w_i\right](k-1)}$$

When the w_i are the true (but unknown) weights, we have

$$\frac{1}{\sum^k w_i} = \frac{\sum^k w_i (\hat{S}_i - \bar{\hat{S}})^2}{\left[\sum^k w_i\right](k-1)}$$

which if we normalize the weights (such that they sum to 1), gives

$$1 = \frac{\sum^k w_i (\hat{S}_i - \bar{\hat{S}})^2}{(k-1)}$$

Since

$$w_i = \frac{1}{\sigma^2 + \mathrm{var}(\hat{S}_i \mid S_i)} \quad \text{and} \quad \bar{\hat{S}} = \frac{\sum^k w_i \hat{S}_i}{\sum^k w_i}$$

then

$$1 = \frac{\sum^k w_i (\hat{S}_i - \bar{\hat{S}})^2}{(k-1)} = \frac{\sum^k \left[\frac{1}{\sigma^2 + \mathrm{var}(\hat{S}_i \mid S_i)} \left(\hat{S}_i - \sum^k \frac{1}{\sigma^2 + \mathrm{var}(\hat{S}_i \mid S_i)} \hat{S}_i\right)^2\right]}{(k-1)}$$

* Gould, W. R. & J. D. Nichols. (1998) Estimation of temporal variability of survival in animal populations. *Ecology*, **79**, 2531-2538.

We then solve (numerically) for $\hat{\sigma}^2$ (which is the only unknown in the expression) – it is convenient to use the naive estimate for σ^2 calculated earlier as a starting point in the numerical optimization.

A confidence interval can be constructed for σ^2 by solving two modified versions of this equation. We assume we want a $(1-\alpha)$% CI, where $\alpha = \alpha_U + \alpha_L$ (where U and L stand for upper and lower, respectively). For the upper limit, we solve for σ^2 in the following

$$\frac{\sum^{k}\left[\frac{1}{\sigma^2+\mathrm{var}(\hat{S}_i \mid S_i)}\left(\hat{S}_i - \sum^{k}\frac{1}{\sigma^2+\mathrm{var}(\hat{S}_i \mid S_i)}\hat{S}_i\right)^2\right]}{(k-1)} = \frac{\chi^2_{k-1,\alpha_L}}{k-1}$$

where χ^2_{n-1,α_L} is the critical value for the central χ^2 distribution corresponding to $(k-1)$ df and α_L percentile.

To find the lower limit, we substitute $\chi^2_{k-1,1-\alpha_U}$ in the RHS of the preceding, and solve for σ^2. If the lower limit does not have a positive solution for σ^2 (since $\sigma \geq 0$), then we set the lower CI to 0 and adjust to a one-sided CI by redefining $\alpha_U = \alpha$.

Burnham *et al*. (1987) describe simplified versions of these estimators for the CI if all the $\mathrm{var}(\hat{S}_i \mid S_i)$ are the same, or nearly so.

D.2. variance components estimation – worked examples

Here, we introduce the 'mechanics' of the variance decomposition, using a series of progressively more complex examples. We begin with a simple example loosely based on a 'known fate' analysis, where survival is estimated as a simple binomial probability, and where there is no covariance among samples.

D.2.1. Binomial survival – simple mean (no sampling covariance)

Imagine a simulated scenario where we are conducting a simple 'known fate' analysis (Chapter 16). In each of 10 years ($k = 10$), we mark and release $n = 25$ individuals, and determine the number alive, y, after 1 year (since this is a known-fate analysis, we assume there is no error in determining whether an animal is 'alive' or 'not alive' on the second sampling occasion). Here, though, we'll assume that the survival probability in each year, S_i, is drawn from $\mathcal{N}(0.5, 0.05)$ (i.e., distributed as an independent normal random variable with mean $\mu = 0.5$ and process variance $\sigma^2 = 0.05^2$). Conditional on each S_i, we generated y_i (number alive after one year in year i) as an independent binomial random variable $B(n, S_i)$. Thus, our ML estimate of survival for each year is $\hat{S}_i = y_i/n$, with a conditional sampling variance of $\widehat{\mathrm{var}}(\hat{S}_i \mid S_i) = [\hat{S}_i(1-\hat{S}_i)]/n$, which given $\mu = 0.5$, and $\sigma^2 = (0.05)^2$ is approximately 0.01.

Table E.1 (top of the next page) gives the values of S_i, y_i and $\hat{S}_i$ for our 'example data'. Clearly, for a 'real analysis', we would not know the true values for S_i – we would have only $\hat{S}_i$, and generally only have $\hat{E}_S(\mathrm{var}(\hat{S}_i \mid S_i))$ as $\widehat{\mathrm{var}}(\hat{S}_i \mid S_i)$.

In Table E.1 we see that the *empirical* standard deviation of the 10 estimated survival rates (i.e., the $\hat{S}_i$) is 0.106. However, we should not take $(0.106)^2$ as an estimate of σ^2 because such an estimate includes *both* process and sampling variation. Clearly, we want to subtract the estimated sampling variance from the total variation to get an estimate of the overall process variation.

Table D.1: *Single realization from simple binomial survival example,* $k = 10, E(S) = 0.5, \sigma = 0.05$*, where* $\hat{S}_i = y_i/n$ *are* $B(25, S_i)$*, hence expected* $SE(\hat{S}_i|S) \approx 0.1$

year (i)	S_i	$\hat{S}_i$	$\widehat{\mathrm{SE}}(\hat{S}_i \mid S_i)$
1	0.603	0.640	0.096
2	0.467	0.360	0.096
3	0.553	0.480	0.100
4	0.458	0.440	0.100
5	0.506	0.480	0.100
6	0.498	0.320	0.093
7	0.545	0.600	0.098
8	0.439	0.400	0.098
9	0.488	0.560	0.099
10	0.480	0.560	0.099
mean	0.504	0.484	0.100
SD	0.050	0.106	

the manual approach...

From section D.1, if we make the strong assumption that all the sampling variances are equal, then the estimate of the overall mean is the mean of the k estimates:

$$\bar{\hat{S}} = \frac{\sum^k \hat{S}_i}{k}$$

with the theoretical variance being

$$\widehat{\mathrm{var}}(\bar{\hat{S}}) = \frac{\sigma^2 + E\left[\mathrm{var}(\hat{S}_i \mid S_i)\right]}{k}$$

In other words the total variance is the sum of the process (environmental) variance, σ^2, plus the expected sampling variance, $E[\mathrm{var}(\hat{S}_i \mid S_i)]$.

From section D.1, and assuming equal weights w_i, where $\sum^k = 1$, we estimate the *total* variance as

$$\widehat{\mathrm{var}}(\bar{\hat{S}}) = \frac{\sum^k (\hat{S}_i - \bar{\hat{S}}_i)^2}{(k-1)}$$

and the expected *sampling* variance as the mean of the sampling variances

$$E\left[\widehat{\mathrm{var}}(\hat{S}_i \mid S_i)\right] = \frac{\sum^k \mathrm{var}(\hat{S}_i \mid S_i)}{k}$$

Thus, we can derive an estimate of the *process* (environmental) variance σ^2 by algebra

$$\hat{\sigma}^2 = \frac{\sum^{10}\left(\hat{S}_i - \bar{\hat{S}}_i\right)^2}{(10-1)} - \frac{\sum^{10}\mathrm{var}\left(\hat{S}_i \,\middle|\, S_i\right)}{10}$$

Thus, from Table (E.1), the process variance for our $k = 10$ samples is estimated as

$$\begin{aligned}\sigma^2 &= \frac{\sum^{10}\left(\hat{S}_i - \bar{\hat{S}}_i\right)^2}{(10-1)} - \frac{\sum^{10}\mathrm{var}\left(\hat{S}_i \,\middle|\, S_i\right)}{10} \\ &= \left(\frac{0.10064}{9}\right) - 0.00959 \\ &= 0.0016 \\ \therefore \sigma &= \sqrt{0.0016} = 0.040\end{aligned}$$

While our estimate of process variance is not much different from the true underlying value (for this example, true $\sigma^2 = 0.0025$), we noted that this naive estimator is not entirely correct, since it assumes equal sampling variances. To obtain an unbiased estimate of σ^2, we weight the sampling variance by

$$w_i = \frac{1}{\sigma^2 + \mathrm{var}(\hat{S}_i \mid S_i)}$$

From section D.1, we derive an estimate of process variance over the $k = 10$ samples by solving (numerically) the following for σ^2

$$\begin{aligned}1 &= \frac{\sum^{10} w_i\left(\hat{S}_i - \bar{\hat{S}}\right)^2}{(10-1)} \\ &= \frac{\sum^{10}\frac{1}{\sigma^2 + \mathrm{var}(\hat{S}_i \mid S_i)}\left(\hat{S}_i - \frac{\sum^{10} w_i \hat{S}_i}{\sum^{10} w_i}\right)^2}{(10-1)} \\ &= \frac{\sum^{10}\left[\frac{1}{\sigma^2 + \mathrm{var}(\hat{S}_i \mid S_i)}\left(\hat{S}_i - \sum^{10}\frac{1}{\sigma^2 + \mathrm{var}(\hat{S}_i|S_i)}\hat{S}_i\right)^2\right]}{(10-1)}\end{aligned}$$

For the present example, our estimated process variance is $\hat{\sigma}^2 = 0.00195$.

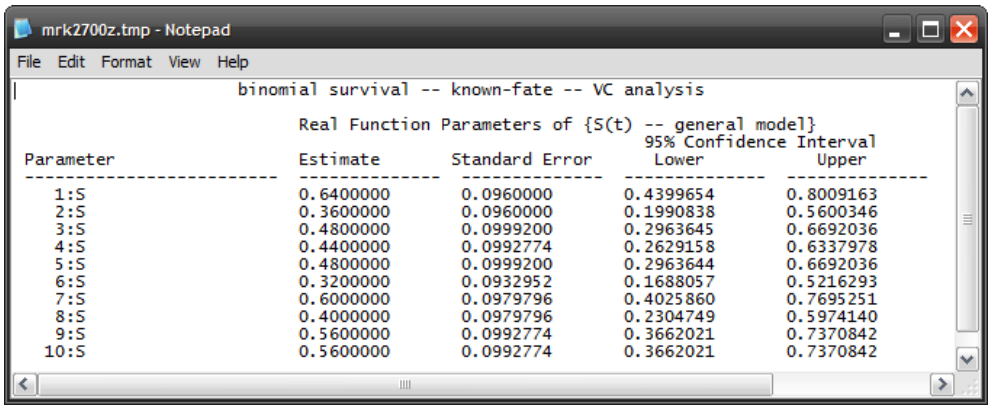
now using MARK...

While estimating process variance 'by hand' is relatively straightforward for this example, we are clearly interested in using the capabilities of program **MARK** to handle the 'heavy lifting' – especially for more complex problems. Here we will introduce some of the 'mechanics' in using program **MARK**

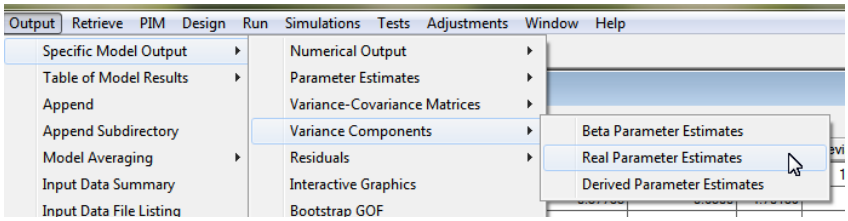
to estimate process variance for our simulated 'known fate' data. The data (number of marked and released animals that survive the one-year interval; see Table E.1) are formatted for the 'known fate' data type, and are contained in `binomial-example.inp`.

Since our purpose here is to demonstrate the mechanics of 'variance components analysis', and not 'known fate analysis', we've gone ahead and built the basic general model $\{S_t\}$ for you. Start **MARK**, and open up `binomial-example.dbf` (*note*: you'll need to have `binomial-example.fpt` installed in the same directory where you have `binomial-example.dbf`). There is only one model in the browser (for now) – model '`S(t) -- general model`'. [At some point, you should look at the underlying PIM structure, to see how we are using the known fate data type to model survival using a simple binomial estimator.]

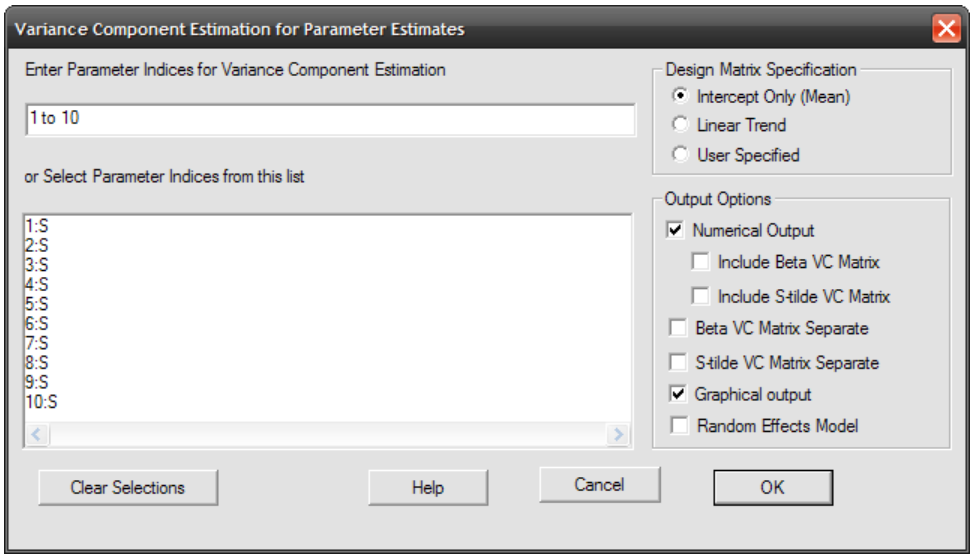
Retrieve model '`S(t) -- general model`', and look at the real parameter estimates (shown below). If you compare these survival estimates with those 'done by hand' in Table E.1, we see they are identical.



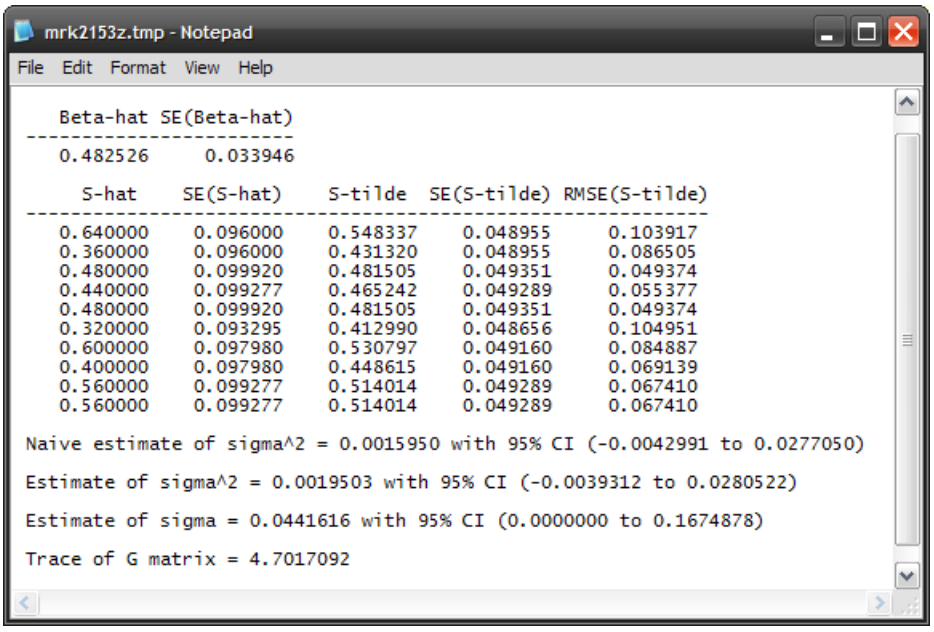
Next, we will use **MARK** to handle the estimation of process variance for us. Start by having a look at the '**`Output | Specific Model Output | Variance Components`**' menu. You'll see that we can execute a variance components analysis on either the β estimates, the real parameter estimates, or on any derived parameter(s). Here, we are interested in the real parameters, so, select that option.



You will then be presented with a window (shown at the top of the next page) asking you to specify which parameters you want to include in the variance component estimation, plus some options concerning the specification of the design matrix, followed by various output options. Note that we specify parameters '`1 to 10`' to be included in the estimation. Since the 'data' were generated based on a sample of survival values drawn from a normal distribution with unknown parametric mean μ and variance σ^2, it should make intuitive sense that we are going to fit a 'mean model (i.e., intercept only). This is checked by default (we will consider other DM specifications elsewhere). For output options, we will accept the default options (selected) – note that at this point, we are not considering the fitting of a 'random effects model' (at least, not directly), so we leave that box unchecked.



Once you have completed entering the parameters and specifying the DM and the output options, click '**OK**'. **MARK** will respond (generally very quickly) by outputting the estimates from the VC analysis into an editor window, shown below (**MARK** will also generate a plot of various estimates – ignore and close the plot for now):



Starting from the top – the first line reports a '`Beta-hat`' of 0.482526. As you might recall from Chapter 6, this is in fact our most robust estimate of the mean survival probability. Note that it is close, but not identical to the simple arithmetic mean $\bar{\tilde{S}}_i = 0.484$. We will outline the reasons for the differences later – for now, we'll accept with deferred proof the statement that '`Beta-hat`' represents our best estimate for mean survival, since it is the estimate of the expected value of S as a random variable. This estimate is followed by the estimate of '`SE(Beta-hat)`'. Here, in the absence of sampling covariance, it is estimated

as the square-root of the sum of estimated process variation, $\hat{\sigma}^2$, and sampling variation, $E\left[\widehat{\mathrm{var}}(\hat{S}_i \mid S_i)\right]$, divided by k, where k is the number of parameter estimates. For our present example, with $k = 10$,

$$\text{'SE(Beta-hat)'} = \sqrt{\frac{\hat{\sigma}^2 + E\left[\widehat{\mathrm{var}}(\hat{S}_i \mid S_i)\right]}{k}}$$
$$= \sqrt{\frac{(0.0019503 + 0.0095872)}{10}} = 0.03396$$

which is what is reported by **MARK** (to within rounding error). Thus, our estimate of total variance would be $(\widehat{\mathrm{SE}}^2 \times k) = (0.03396^2 \times 10) = 0.01153$ (i.e., the value of the numerator inside the square-root, above).

Next, a table of various parameter estimates. The first two columns should be self-explanatory – the ML estimates of survival $\hat{S}_i$ ('S-hat'), followed by the binomial standard error for the estimate ('SE(S-hat)'. Next, the 'shrinkage' estimates $\tilde{S}_i$ ('S-tilde') and their corresponding SE and RMSE. The derivation, use and interpretation of the shrinkage estimates is developed in section D.3.

Finally, the estimates for process variation (the **G** matrix we'll get to later). First, **MARK** reports the 'Naive estimate of sigma^2'= 0.001595. This is *exactly* the same value as the 'first approximation' we derived 'by hand' in the preceding section. This is followed by the 'Estimate of sigma^2' = 0.0019503 (and the 'Estimate of sigma' = 0.044162). Both estimates are *identical* to those we derived 'by hand' using the 'weighted means' approach in the preceding section.

D.2.2. Binomial example extended – simple trend

Here we consider a similar scenario, again involving a simple 'known fate' analysis with no sampling covariance among samples. In each of 15 years ($k = 15$), we mark and release $n = 25$ individuals, and determine the number alive, y, after 1 year. Here, though, we assume that the true mean survival probability in each year, $\bar{S}_i$, is declining over time (from 0.60 in the first year, to 0.46 in the final year). We'll assume that the survival probability in each year, S_i, is drawn from $\mathcal{N}(\bar{S}_i, 0.05)$. Conditional on each S_i, we generated y_i (number alive after one year in year i) as an independent binomial random variable $B(n, S_i)$. Table D.2 (top of the next page) gives the values of S_i, y_i and $\hat{S}_i$ for our 'example data'.

Now, in the preceding example, the survival probability in each year, S_i, was drawn from $\mathcal{N}(0.5, 0.05)$ (i.e., distributed as an independent normal random variable with mean $\mu = 0.5$ and *process* variance $\sigma^2 = 0.05^2$). Here, though, not only is there random variation around the mean, but the mean itself declines over time. In this example there are 2 sources of process variation: the random variation around the mean survival in any given year, and the decline over time in the value of the mean. As such, we anticipate that the actual process variance will be $> (0.05)^2$.* We also anticipate that if we estimate process variance using a model based on a simple mean (i.e., where we assume that process variation is due only to variation around a mean survival which is the same in all years) the estimate will be biased high (since it will conflate within- and among-year sources of variation). What about sampling variation? The imposition of a trend does not influence sampling variation – in each year, sampling is based on a binomial with the same number of individuals 'released' each year.

We'll test both expectations, using data contained in `binomial-example-trend.inp`. Again, we've provided you with some 'pre-built' models to start with (contained in `binomial-example-trend.dbf` and `binomial-example-trend.fpt`). We'll avoid doing the same 'hand calculations' we worked through

* The actual value of the process variance, accounting for both within and among season variation, is $\sigma^2 = 0.0045$. We'll leave it to you as an exercise to derive this value yourself.

Table D.2: *Single realization from simple binomial survival example,* $k = 15$*, S declining linearly from* $0.6 \rightarrow 0.46$ *(*$\bar{S} = 0.53$*),* $\sigma = 0.05$*, where* $\hat{S}_i = y_i/n$ *are* $B(25, S_i)$ *– hence expected* $SE(\hat{S}_i \mid S) \approx 0.1$*.*

year (i)	S_i	$\hat{S}_i$	$\widehat{\mathrm{SE}}(\hat{S}_i \mid S_i)$
1	0.647	0.560	0.0099
2	0.595	0.440	0.0099
3	0.667	0.440	0.0099
4	0.580	0.640	0.0092
5	0.532	0.640	0.0092
6	0.475	0.720	0.0081
7	0.624	0.360	0.0092
8	0.516	0.400	0.0096
9	0.640	0.520	0.0100
10	0.430	0.480	0.0100
11	0.503	0.400	0.0096
12	0.509	0.520	0.0100
13	0.533	0.360	0.0092
14	0.394	0.360	0.0092
15	0.490	0.240	0.0073
mean	0.542	0.472	0.0093
SD	0.081	0.129	

in the preceding example (same basic idea, but a fair bit messier because of having to account for both within and among year variation), and simply use **MARK**.

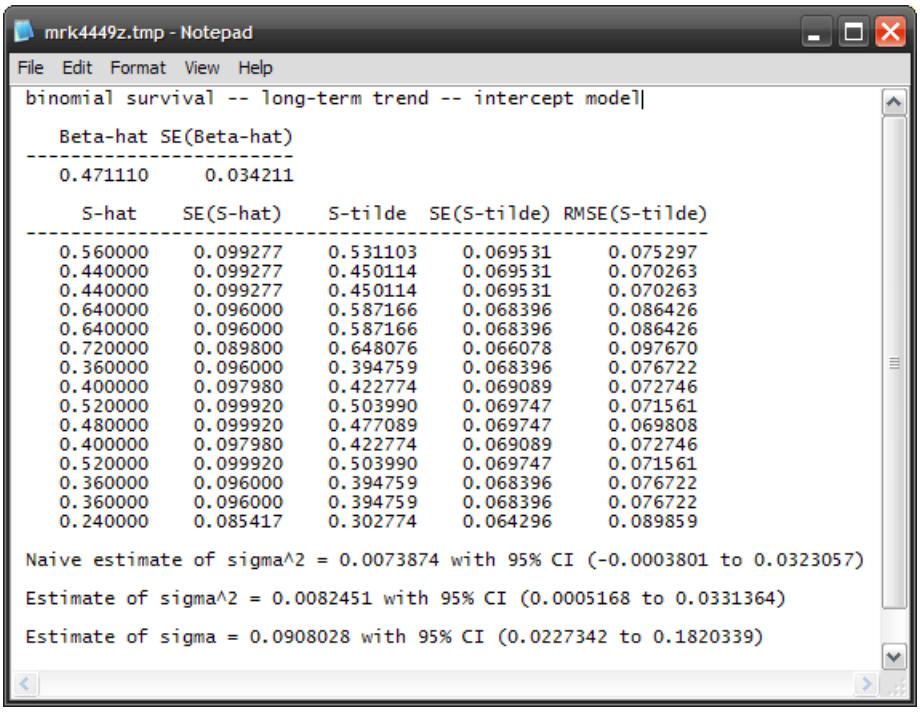
Start up **MARK**, and open up `binomial-survival-trend.dbf`. You'll see that there are 2 'pre-built' models in the browser: '`S(t) - DM`' (simple time variation) and '`S(T) - DM`' (a trend model, where annual estimates are constrained to follow a linear trend). The '`-DM`' simply indicates that both were constructed using a design matrix. Based on AIC_c weights, there is clearly far more support for the trend model than the model with simple time variation. This is a good thing, since the data being analyzed were simulated under a trend model.

Our purpose here, though, is not to do 'model selection' (we'll get there). Our present interest is on estimating the variance components. So, first question. Which model do we want to estimate variance components from? This is a more subtle question than it might seem. On the one hand, if we didn't know there was a trend, it might seem that we should select the time-dependent model since it is more general. On the other hand, you might have prior information suggesting a trend, and might think that it is a better model. Or, you might build both models, see that the trend model has the most support in the data, and use that model as the basis for estimating variance components. You need to think this through carefully. We are trying to estimate process variance – we want to estimate the magnitude of the joint within- and among-year variation in our data. Thus, we want to use the most general model possible. In this case, model '`S(t) - DM`'. We don't use model '`S(T) - DM`', since the estimates from that model are constrained to fall on a straight line. Meaning, the only remaining variation would be the annual variation in mean survival (as estimated by the regression equation). Meaning that any estimate of process variation from such a model would massively underestimate true process variation in the data. So, start by retrieving model '`S(t) - DM`'. Then, select '**`Output | Specific model output | Variance components | Real parameter estimates`**'. With 15 samples (occasions) we specify '`1 to 15`'.

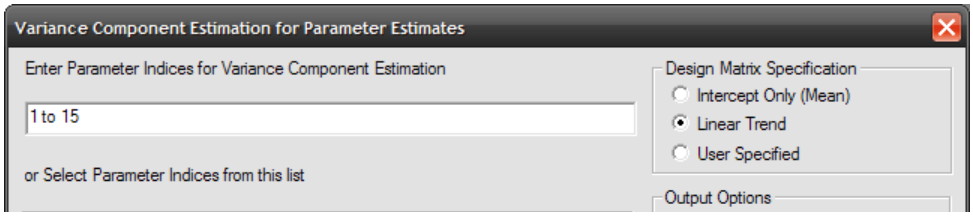
What about the 'design matrix specification'? Recall from the preceding example that we used the default '**Intercept Only (mean)**' specification. However, there are 2 other options available to you: '**linear trend**', and '**user specified**'. In effect, the first 2 options ('**intercept only**' and '**linear trend**') are there simply for your convenience, since both models are very commonly used. You could, however, build either model by selecting the '**user specified**' option (which essentially is the option you select if you want to build a specific model directly, using the design matrix). We'll defer using the '**user specified**' option for now, and simply compare the '**intercept only**' and '**linear trend**' models. We'll start with the default '**intercept only**' option.

Once you click the '**OK**' button, **MARK** will respond with the estimates of year-specific survival probabilities, and the estimates of total and process variance (shown at the top of the next page). Again, the first line is the estimate of the overall mean, $\hat{\bar{S}} = 0.4711$, and SE = 0.0342 (representing total variance). Note that the reported mean is very close to the mean of the true S_i (Table D.2), $\bar{S}_i = 0.472$.

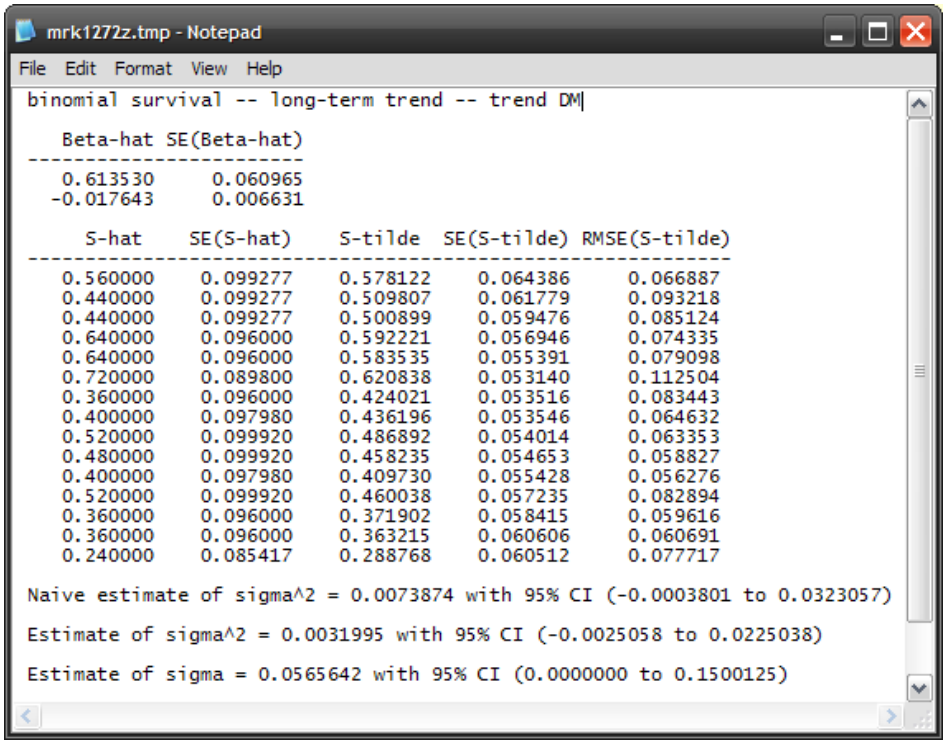
What about the estimated variance? The estimate of process variance $\hat{\sigma}^2 = 0.00825$, which as we anticipated is almost twice as large as the true process variance in the data ($\sigma^2 = 0.0045$). Thus, the estimated SE for total variance will also be biased high.



Now, let's do a variance components analysis on the time-dependent model, by checking the '**linear trend**' DM option, as shown below:



The parameter estimates are shown at the top of the next page. Note that we no longer have an estimate of a single 'Beta-hat' (i.e., we no longer have an estimate of just the mean). What we do have, though, is an estimate of the intercept ($\hat{\beta}_1 = 0.61353$), and the slope of the decline over time ($\hat{\beta}_2 = -0.017643$). Both are quite close to the values of $\hat{\beta}_1 = 0.64$ and $\hat{\beta}_2 = -0.0102$, estimated from a regression of the true S_i (Table D.2) on year. This is not surprising. The estimated process variance, $\hat{\sigma}_2 = 0.0031995$, is much more consistent with the true process variance, $\sigma^2 = 0.0045$, than was our estimate under the 'intercept only' model.



D.2.3. what about sampling covariance?

In the preceding examples, we considered situations where there was no sampling covariance. While this is a useful place to start, it is not particularly realistic in many situations, where sampling covariances are potentially not small. Recall that our simple (naive) estimator for process variance, $\hat{\sigma}^2$, for some parameter θ was given as

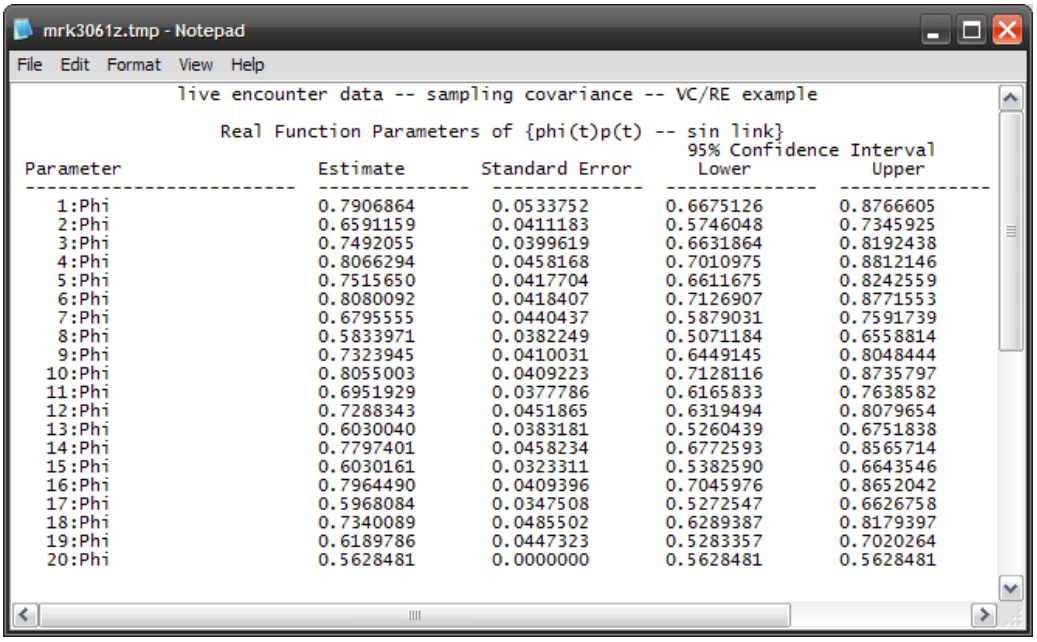
$$\hat{\sigma}^2 = \frac{\sum^{k-1}(\hat{\theta}_i - \bar{\hat{\theta}})^2}{k-1} - E\left[\mathrm{var}(\hat{\theta}_i \mid S_i)\right]$$

This is a suitable *first* approximation when sampling covariances are 0, or nearly so.

However, when sampling covariances are significant, then we need to modify the estimator for σ^2 to explicitly account for the sampling covariance.

$$\hat{\sigma}^2 = \frac{\sum^{k-1}(\hat{\theta}_i - \bar{\hat{\theta}})^2}{k-1} - E\left[\mathrm{var}(\hat{\theta}_i \mid \theta_i)\right] + E\left[\mathrm{cov}(\hat{\theta}_i, \hat{\theta}_j \mid \theta_i, \theta_j)\right]$$

To illustrate how sampling covariance is handled, we simulated a data set of live-encounter (CJS) data, 21 occasions, 350 newly marked individuals released at each occasion, under true model $\{\varphi_t p_t\}$. Apparent survival, φ_i, over a given interval i was generated by selecting a random beta deviate drawn from $\mathcal{B}(0.7, 0.005)$. The encounter probability p_i at a sampling occasion i was generated by selecting a random beta deviate drawn from $\mathcal{B}(0.35, 0.005)$. The simulated live-encounter data are contained in `normsim-VC.inp`. The estimates $\hat{\varphi}_i$ from model $\{\varphi_t p_t\}$ are shown at the top of the next page. For a time-dependent model, the terminal φ and p parameter estimates are confounded (reflected in the estimated $\widehat{\mathrm{SE}}(\hat{\varphi}_{20}) = 0.000$). This becomes important when we specify the parameters in the variance-components analysis.



mrk3061z.tmp - Notepad

File Edit Format View Help

live encounter data -- sampling covariance -- VC/RE example

Real Function Parameters of {phi(t)p(t) -- sin link}

Parameter	Estimate	Standard Error	95% Confidence Interval Lower	Upper
1:Phi	0.7906864	0.0533752	0.6675126	0.8766605
2:Phi	0.6591159	0.0411183	0.5746048	0.7345925
3:Phi	0.7492055	0.0399619	0.6631864	0.8192438
4:Phi	0.8066294	0.0458168	0.7010975	0.8812146
5:Phi	0.7515650	0.0417704	0.6611675	0.8242559
6:Phi	0.8080092	0.0418407	0.7126907	0.8771553
7:Phi	0.6795555	0.0440437	0.5879031	0.7591739
8:Phi	0.5833971	0.0382249	0.5071184	0.6558814
9:Phi	0.7323945	0.0410031	0.6449145	0.8048444
10:Phi	0.8055003	0.0409223	0.7128116	0.8735797
11:Phi	0.6951929	0.0377786	0.6165833	0.7638582
12:Phi	0.7288343	0.0451865	0.6319494	0.8079654
13:Phi	0.6030040	0.0383181	0.5260439	0.6751838
14:Phi	0.7797401	0.0458234	0.6772593	0.8565714
15:Phi	0.6030161	0.0323311	0.5382590	0.6643546
16:Phi	0.7964490	0.0409396	0.7045976	0.8652042
17:Phi	0.5968084	0.0347508	0.5272547	0.6626758
18:Phi	0.7340089	0.0485502	0.6289387	0.8179397
19:Phi	0.6189786	0.0447323	0.5283357	0.7020264
20:Phi	0.5628481	0.0000000	0.5628481	0.5628481

Estimation of σ^2 under the naive model is straightforward. The only additional complications are that (i) the terms in the estimator are calculated over $\varphi_1 \rightarrow \varphi_{19}$, and (ii) we have to calculate the mean of the sample covariances to estimate $E\left[\mathrm{cov}(\hat{\varphi}_i, \hat{\varphi}_j \mid \varphi_i, \varphi_j)\right]$. In practice, this second step isn't too difficult, depending on your facility with computers. You simply need to find a way to calculate the mean of the off-diagonal elements of the V-C matrix (keeping in mind you're calculating the mean over $\varphi_1 \rightarrow \varphi_{19}$). For the present example, $\widehat{\mathrm{cov}}(\hat{\varphi}_i, \hat{\varphi}_j \mid \varphi_i, \varphi_j) = -0.00008$. Thus,

$$\begin{aligned}
\hat{\sigma}^2 &= \frac{\sum_{k-1}^{k-1}(\hat{\varphi}_i - \bar{\hat{\varphi}})^2}{k-1} - E\left[\mathrm{var}(\hat{\varphi}_i \mid \varphi_i)\right] + E\left[\mathrm{cov}(\hat{\varphi}_i, \hat{\varphi}_j \mid \varphi_i, \varphi_j)\right] \\
&= \left(\frac{0.11531}{18}\right) - \left(\frac{0.0338}{19}\right) - 0.00008 \\
&= 0.004552
\end{aligned}$$

Clearly, the proportional contribution of the covariance term is very small (2%). This will often be the case, especially for time-dependent models.

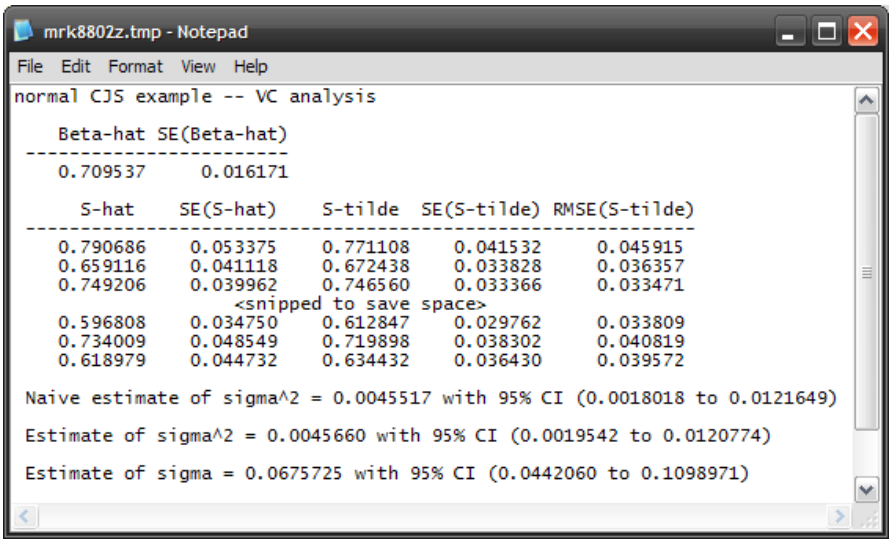
If we analyze these live-encounter data using the variance components routines in **MARK**, using

the '**intercept only**' mean model, the reported value for the 'naive' estimate (shown at the top of the next page) matches the value we derived by hand on the preceding page. The estimate based on the 'weighted' estimator is almost identical – and both are not too far off the true value of $\sigma^2 = 0.005$.

The near equivalence of the 'naive' and 'weighted' estimates reflects the fact that sampling variation is small, relative to process variance, in this example (small sampling variance is not surprising, given that the data were generated under a model with $p = 0.35$ and 350 individuals marked and released on each sampling occasion). Recall from section D.1 that from least-squares theory, we should weight our estimates of total and sampling variance to obtain an unbiased estimate of process variance, σ^2, using a weight w_i

$$w_i = \frac{1}{\sigma^2 + \mathrm{var}(\hat{\varphi}_i \mid \varphi_i)}$$

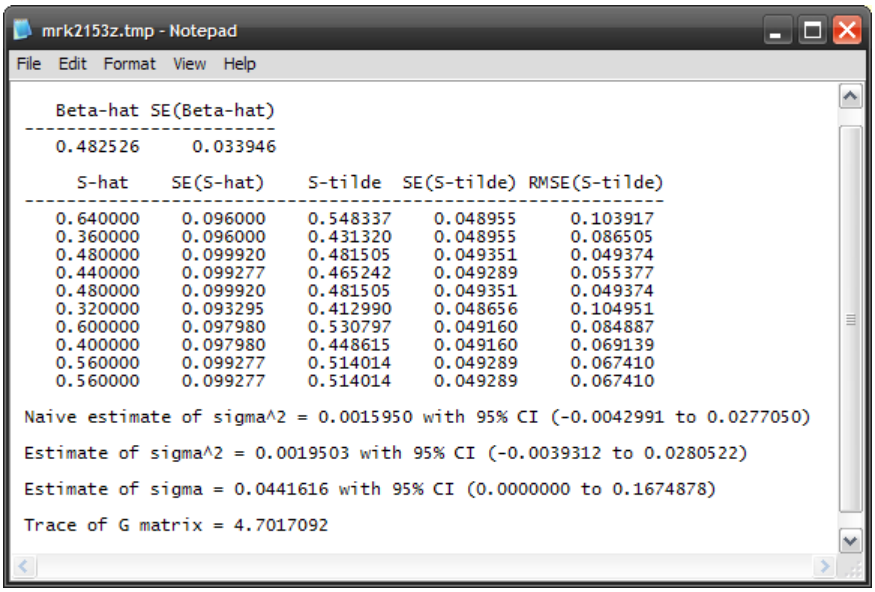
For this example, $\mathrm{var}(\hat{\varphi}_i \mid \varphi_i) \ll \sigma^2, \forall i$, and so $w_i \approx 1/\sigma^2$, which is a constant (since σ^2 is a constant). Thus, for this example, the weighting does not change the naive estimate.



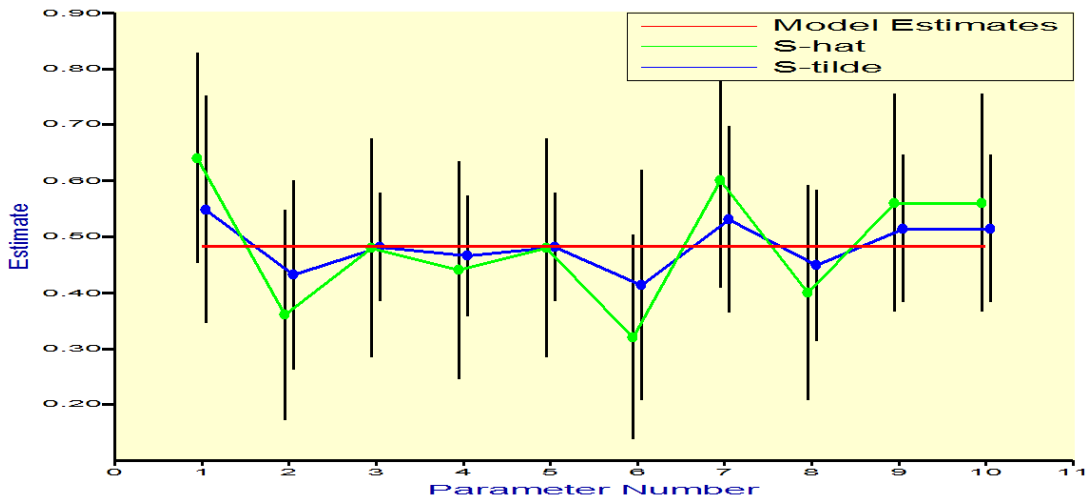
You may have noticed that the ML estimates ('S-hat') are very close to what we identified earlier as 'shrinkage' estimates ('S-tilde'). Is the near-equivalence of the 'naive' and 'weighted' estimates for σ^2 related to the 'closeness' of the ML and 'shrinkage' estimates?

D.3. random effects models and shrinkage estimates

In this section, we introduce what we will refer to as 'random effects' models. We'll begin by having another look at the results from the simple binomial example (section D.2.1), shown on the top of the next page. From left to right are the ML estimates, $\hat{S}_i$ ('S-hat'), the estimated standard error for the ML estimate, $\widehat{\mathrm{SE}}(\hat{S}_i \mid S)$ ('SE(S-hat)'), the corresponding 'shrinkage' estimate, $\tilde{S}_i$ ('S-tilde'), the estimated standard error for the shrinkage estimate, $\widehat{\mathrm{SE}}(\tilde{S}_i \mid \hat{S}_i)$, and the estimated residual mean-squared error (RMSE) for the shrinkage estimate, $\widehat{\mathrm{RMSE}}(\tilde{S}_i \mid \hat{S}_i)$ ('RMSE(S-tilde)').



Here is a plot of the ML estimates, $\hat{S}_i$ (green line), the 'shrinkage' estimates, $\tilde{S}_i$ (blue line), and the model estimates (for the mean model, corresponding to the estimated mean $\hat{\beta} = 0.4825$; red line).



We are familiar with the 'ideas' behind the ML estimates, $\hat{S}_i$, and the idea of an overall estimate of the mean survival, $\hat{E}(S)$, hopefully makes some intuitive sense. But, what are 'shrinkage' estimates? We'll start with a short-form explanation, focussing on the basic ideas, then jump down into the weeds a bit for a deeper (more technical) discussion. The concept of a 'shrinkage' estimate is not the easiest thing to understand.* We will follow this by illustrating the mechanics of building and fitting these models in **MARK**, through a series of 'worked examples'.

* In his text on matrix population models, Hal Caswell comments that understanding eigenvalues and eigenvectors (which feature prominently in demographic analysis) requires 'not only a mechanical understanding, but a real intuitive grasp of the slippery little suckers...' (p. 662, 2nd edition). We submit the same is true for 'shrinkage' estimates.

D.3.1. the basic ideas...

Looking at the tabular output and the plot, there are notable differences in point estimates, and precision, between the the ML estimates and the shrinkage estimates. If you look carefully, you'll notice that for most years, the shrinkage estimate falls somewhere between the ML estimate, and the mean. The shrinkage method is so called because each residual arising from the fitted reduced-parameter model ($\hat{S}_i - \hat{E}(S)$) is 'shrunken', then added back to the estimated model structure for observation i under that reduced model. In a heuristic sense, the $\tilde{S}_i$ are derived from the ML estimates by removal of the sampling variation.

When sampling covariances are zero*, the shrinkage estimator used in **MARK** for the mean-only model (although the structure applies generally) is

$$\tilde{S}_i = \hat{E}(S) + \sqrt{\frac{\hat{\sigma}^2}{\hat{\sigma}^2 + \hat{E}_S\left[\widehat{\mathrm{var}}(\hat{S}_i \mid S_i)\right]}} \times \left[\hat{S}_i - \hat{E}(S)\right]$$

The first term on the right-hand side (RHS) of the expression is the estimate of the mean survival, $\hat{E}(S)$. The last term, $[\hat{S}_i - \hat{E}(S)]$, is simply the residual of the ML estimate from the model. But what about the middle term on the RHS?

We will generally refer to

$$\sqrt{\frac{\hat{\sigma}^2}{\hat{\sigma}^2 + \hat{E}_S\left[\widehat{\mathrm{var}}(\hat{S}_i \mid S_i)\right]}}$$

as the 'shrinkage coefficient', and it clearly is a function of the proportion of total variance – i.e., the sum $\sigma^2 + \mathrm{var}(\hat{S}_i \mid S_i)$ – due to process (environmental) variation, σ^2. The square-root is used because then

$$\hat{\sigma}^2 \doteq \frac{\sum^k (\tilde{S}_i - \bar{\tilde{S}})^2}{k-1} \quad \text{and} \quad \hat{E}(S) \doteq \bar{\tilde{S}}$$

If there is no process variation (i.e., $\sigma^2 = 0$), then the shrinkage coefficient is evaluated at 0, and thus the shrinkage estimate would be the mean, $\hat{E}(S)$. In other words, if you have no environmental variation, then the only variation in the system is sampling. If you remove the sampling variation (which we noted earlier is what shrinkage is doing, at least heuristically), then this makes sense – without process or sampling variation, every shrinkage estimate would simply be the mean, i.e., $\tilde{S}_i = \hat{E}(S)$. In contrast, with increasing process variance (i.e., increasing σ^2), we see that as $\sigma^2 > \mathrm{var}(\hat{S}_i \mid S_i)$ (i.e., as the proportion of total variance due to process variance increases), then the shrinkage coefficient approaches 1.0, and the shrinkage estimate would approach the ML estimate, $\hat{S}_i$.

This relationship is depicted in the following diagram where the arrows indicate the direction that decreasing or increasing process variance σ^2 has on the value of the shrinkage estimate, $\tilde{S}_i$, relative to the arithmetic average of the ML estimate $\hat{S}_i$ and the mean $\hat{E}(S)$.

$$\hat{E}(S) \xleftarrow[\downarrow \sigma^2]{} \tilde{S}_i \xrightarrow[}{\uparrow \sigma^2} \hat{S}_i$$

Thus, another way of looking at it is to view the shrinkage estimate is analogous to an 'average'

* The full shrinkage estimator, accounting for non-zero sampling covariances, is presented in section D.3.2.

between the two estimates. This is important when we consider model averaging – we defer discussion of that important topic until section D.5.

We should note that a shrinkage coefficient different than

$$\sqrt{\frac{\hat{\sigma}^2}{\hat{\sigma}^2 + \hat{E}_S\left[\widehat{\mathrm{var}}(\hat{S}_i \mid S_i)\right]}}$$

could be used. However, this particular shrinkage coefficient has a very desirable property: if we treat the $\tilde{S}_i$ as if they were a random sample, then their sample variance almost exactly equals $\hat{\sigma}^2$. This also means that a plot of the shrinkage residuals (as implicit in the plot at the top of the preceding page) gives a correct visual image of the process variation in the S_i.

This starts to point us in the direction of answering the basic question 'why derive shrinkage estimators for the S_i?'. The answer in part comes from the observation we just made that the sample variance of the shrinkage estimates is very close to the estimate of process variance, $\hat{\sigma}^2$. So, the shrinkage estimates are 'improved', relative to the ML estimates, since they should have better precision. If we look at the estimates for the binomial survival example, we see that the improvements gained by the shrinkage estimators, $\tilde{S}_i$, appears substantial – they have about 50% better precision (simply compare $\widehat{\mathrm{SE}}(\tilde{S}_i \mid S_i)$ to $\widehat{\mathrm{SE}}(\hat{S}_i \mid S_i)$).

However, because the ML estimates are unbiased, and the shrinkage estimators are biased (as we will explain), a necessary basis for a fair comparison is the sum of squared errors (SSE). The SSE is a natural measure of the closeness of a set of estimates to the set of S_i. For example, for the binomial survival example, the SSE for the ML estimates is

$$\mathrm{SSE}_{MLE} = \sum^{10} (\hat{S}_i - S_i)^2 = 0.067$$

while for the shrinkage estimates, the SSE is

$$\mathrm{SSE}_{shrinkage} = \sum^{10} (\tilde{S}_i - S_i)^2 = 0.019$$

Clearly, in this sample the shrinkage estimates, as a set, are closer to truth. The expected SSE is the mean square error, MSE (= E[SSE]), which is a measure of average estimator performance.

Those of you with some background in statistical theory might see the connections between the preceding and James-Stein estimation, wherein (in highly simplified form) when 3 or more parameters are estimated simultaneously, there exist combined estimators more accurate on average (that is, having lower expected MSE) than any method that considers the parameters separately. For example, let $\boldsymbol{\theta}$ is a vector consisting of $n \geq 3$ unknown parameters. To estimate these parameters, we take a single measurement X_i for each parameter θ_i, resulting in a vector $\mathbf{X}$ of length n. Suppose the measurements are independent, Gaussian random variables, such that $\mathbf{X} \sim \mathcal{N}(\mu, 1)$. The most obvious approach to parameter estimation would be to use each measurement as an estimate of its corresponding parameter: $\hat{\boldsymbol{\theta}} = \mathbf{X}$. James-Stein demonstrated that this standard (LS) estimator is suboptimal in terms of mean squared error, $E(\boldsymbol{\theta} - \hat{\boldsymbol{\theta}})$. In other words, there exist alternative estimators which always achieve lower mean squared error, no matter what the value of $\boldsymbol{\theta}$ is. For example, it can be shown that a combined estimator of the sample and global mean is a better predictor of the future than is the individual sample mean, since the total MSE of the combined estimator is lower than if using the sample means themselves. This clearly points to part of the theory underlying the use of shrinkage estimators – James-Stein says

that a combined estimator (say, of the ML estimate and the random mean, which is of course our shrinkage estimate) will have a lower MSE than will the ML estimates themselves (with the degree of improvement increasing with increasing number of sample means being combined).

It is important to note, however, that the combined estimator will be closer to optimal *overall*, since it minimizes the MSE of the estimates overall. However, it is possible that any one individual estimate could be 'incorrectly shrunk' (relative to the true value of the parameter), even in the wrong direction. So, shrinkage is conceptually optimal for the set of parameters, but not necessarily for any individual parameter).

If these ideas are new to you, papers by Efron & Morris (1975, Data analysis using Stein's Estimator and its generalizations, *JASA* **70**: 311-319; 1977, Stein's paradox in statistics, *Scientific American* **238**: 119-127) are quite accessible, and provide excellent introductions to the subject.

begin sidebar

shrinkage estimators and 95% confidence limits

For the ML estimates, an approximate 95% confidence interval on S_i is given by $\hat{S}_i \pm 2\widehat{\mathrm{SE}}(\hat{S}_i \mid S_i)$. This procedure will have good coverage in this example. However, for the shrinkage estimator if we use $\tilde{S}_i \pm 2\widehat{\mathrm{SE}}(\tilde{S}_i \mid S_i)$, coverage will be negatively affected by the bias of $\tilde{S}$. Theory (discussed in B&W) shows that the correct expected coverage occurs for the interval $\tilde{S}_i \pm 2\widehat{\mathrm{RSME}}(\tilde{S}_i \mid S_i)$, where

$$\widehat{\mathrm{RSME}}(\tilde{S}_i \mid S_i) = \sqrt{\widehat{\mathrm{var}}(\tilde{S}_i \mid S_i) + (\tilde{S}_i - \hat{S}_i)^2}$$

The expectation over S_i of $\widehat{\mathrm{MSE}}(\tilde{S}_i \mid S_i) = [\widehat{\mathrm{RSME}}(\tilde{S}_i \mid S_i)]^2$ is approximately the mean square error for $\tilde{S}_i$, MSE_i. For the ML estimates,

$$\widehat{\mathrm{RSME}}(\hat{S}_i \mid S_i) = \widehat{\mathrm{SE}}(\hat{S}_i \mid S_i)$$

because $\hat{S}_i$ is unbiased. The unbiasedness of the ML estimates in the general model, together with a high correlation between $\hat{S}_i$ and $\tilde{S}_i$, and the assumption that S_i are random, allows an argument that $\widehat{\mathrm{RMSE}}(\tilde{S}_i \mid S_i)$ is an estimator of the unconditional sampling standard error of $\tilde{S}_i$ (over conceptual repetitions of the data). It then follows that this RMSE can be a correct basis for a reliable confidence interval. It is rare to have a reliable estimator of the MSE for a biased estimator, but when this occurs it makes sense to use $\pm 2\sqrt{\widehat{\mathrm{MSE}}}$ rather than $\pm 2\widehat{\mathrm{SE}}$, as the basis for a 95% CI.

end sidebar

D.3.2. some technical background...

Most random effects theory assumes conditional independence of the estimators. In the introduction to this section, we started by having another look at the simple binomial survival example introduced earlier in section D.2.1. In that example, there was no sampling covariance – the estimates were all independent of each other.

However, more generally in capture-recapture studies, the estimators $\hat{S}_1, \ldots, \hat{S}_k$ are pairwise conditionally correlated. Thus, a more general, extended theory is required, which we develop here in summary form – complete details are found in B&W. While some of the math can get a bit ugly, some familiarity with the ideas (at least) is helpful in more fully understanding 'what **MARK** is doing'. In the following, vectors (all column) are underlined. Matrices are in bold font. A matrix, **X**, may be a vector if it has only a single column. In that case, we do not underline **X**.

First, we assume $\underline{\hat{S}} = \underline{S} + \underline{\delta}$, given $\underline{S}$. Conditional on $\underline{S}$, $\underline{\delta}$ (which has a zero expectation) has a variance-covariance matrix **W**, and $E(\underline{\hat{S}} \mid \underline{S}) = \underline{S}$ for large samples. Second, unconditionally $\underline{S}$ is a random vector with expectation $\mathbf{X}\underline{\beta}$ and variance-covariance matrix $\sigma^2\mathbf{I}$, where **I** is the identity matrix. (Note: the vector $\underline{\beta}$ is different than the beta parameters of the **MARK** link function). Thus, the process errors $\epsilon_i = S_i - E(S_i)$ are independent with homogeneous variance σ^2. Also, we assume mutual independence of sampling errors δ and process errors ϵ. We fit a model that does not constrain S, e.g., $\{S_t\}$, and hence get the maximum likelihood estimates $\underline{\hat{S}}$ and an estimate of **W**.

Let $\underline{S}$ be a vector with n elements, and $\underline{\beta}$ have k elements. Unconditionally,

$$\underline{\hat{S}} = \mathbf{X}\underline{\beta} + \underline{\delta} + \underline{\epsilon}$$
$$\mathrm{VC}(\underline{\delta} + \underline{\epsilon}) = \mathbf{D} = \sigma^2\mathbf{I} + \mathbf{W}$$

We want to estimate $\underline{\beta}$ and σ^2, an unconditional variance-covariance matrix for $\underline{\hat{\beta}}$, a confidence interval on σ^2, and to compute a shrinkage estimator of S (i.e., $\tilde{S}$) and its conditional sampling variance-covariance matrix. In this random effects context the maximum likelihood estimator is the best conditional estimator of $\underline{S}$. However, once we add the random effects structure we can consider an unconditional estimator of $\underline{S}$ ($\tilde{S}$) and a corresponding unconditional variance-covariance for $\tilde{S}$, which incorporates σ^2 as well as **W** and has $(n - k)$ degrees of freedom (if we are assuming large df for **W** and 'large' σ^2).

For a given σ^2 we have

$$\underline{\hat{\beta}} = (\mathbf{X}'\mathbf{D}^{-1}\mathbf{X})^{-1}\mathbf{X}'\mathbf{D}^{-1}\underline{\hat{S}}$$

Note that here **D**, hence $\underline{\hat{\beta}}$, is a function of σ^2. Now we need a criterion that allows us to find an estimator of σ^2. Assuming normality of $\underline{\hat{S}}$ (approximate normality usually suffices), then the weighted residual sum of squares $(\underline{\hat{S}} - \mathbf{X}\underline{\hat{\beta}})'\mathbf{D}^{-1}\underline{\hat{S}} - \mathbf{X}\underline{\hat{\beta}})$ is central χ^2-distributed on $(n - k)$ degrees of freedom. Hence, by the method of moments*

$$n - k = (\underline{\hat{S}} - \mathbf{X}\underline{\hat{\beta}})'\mathbf{D}^{-1}(\underline{\hat{S}} - \mathbf{X}\underline{\hat{\beta}})$$

This equation defines a 1-dimensional numerical-solution search problem. Pick an initial (starting) value of σ^2, compute **D**, then compute $\underline{\hat{\beta}}$, then compute the right-hand side of the preceding expression. This process is repeated over values of σ^2 until the solution, as $\underline{\hat{S}}$, is found. This process also gives $\underline{\hat{\beta}}$.

The unconditional variance-covariance matrix of $\underline{\hat{\beta}}$ is $\mathrm{VC}(\underline{\hat{\beta}}) = (\mathbf{X}'\mathbf{D}^{-1}\mathbf{X})^{-1}$.

Now we define another matrix as

$$\mathbf{H} = \sigma\sqrt{\mathbf{D}}$$
$$= \sigma\sqrt{\sigma^2\mathbf{I} + E(\mathbf{W})}$$
$$= \sqrt{\mathbf{I} + \frac{1}{\sigma^2}E(\mathbf{W})}$$

(Here we only need **H** at $\hat{\sigma}$.)

* Which is why the random effects estimation procedure in **MARK** is sometimes referred to as 'the moments estimator'.

The recommended shrinkage estimate (which is what is used in **MARK**) is

$$\begin{aligned}\underline{\tilde{S}} &= \mathbf{H}(\underline{\hat{S}} - \mathbf{X}\underline{\hat{\beta}}) + \mathbf{X}\underline{\hat{\beta}} \\ &= \mathbf{H}\underline{\hat{S}} + (\mathbf{I} - \mathbf{H})\mathbf{X}\underline{\hat{\beta}}\end{aligned}$$

To get an estimator of the conditional variance of these shrinkage estimators (which is not exact as the estimation of σ^2 is ignored here, as it is for the variance-covariance matrix of $\underline{\hat{\beta}}$), we define and compute a projection matrix **G** as follows:

$$\mathbf{G} = \mathbf{H} + (\mathbf{I} - \mathbf{H})\mathbf{A}\mathbf{D}^{-1}$$

Hence,

$$\underline{\tilde{S}} = \mathbf{G}\underline{\hat{S}}$$

In other words, **G** is the projection matrix which 'maps' the vector of ML estimates to the vector of shrinkage estimates.

The conditional variance-covariance matrix of the shrinkage estimator is then $\mathrm{VC}(\underline{\tilde{S}} \mid \underline{S}) = \mathbf{GWG}'$, whereas $\mathbf{W} = \mathrm{VC}(\underline{\tilde{S}} \mid \underline{S})$. Because $\underline{\tilde{S}}$ is known to be biased, and because the direction of the bias is known, an improved basis for inference is $\mathrm{VC}(\underline{\tilde{S}} \mid \underline{S}) = \mathbf{GWG}' + (\underline{\tilde{S}} - \underline{\hat{S}})(\underline{\tilde{S}} - \underline{\hat{S}})'$. The square-roots of the diagonal elements of this matrix are

$$\widehat{\mathrm{RMSE}}(\tilde{S}_i \mid \underline{S}) = \sqrt{\widehat{\mathrm{var}}(\tilde{S}_i \mid \underline{S}) + (\tilde{S}_i - \hat{S}_i)^2}.$$

As discussed earlier, confidence intervals should be based on this RMSE. The RMSE can exceed the $\mathrm{SE}(\hat{S}_i \mid S_i)$, but on average, the RMSE is smaller.

D.3.3. Deriving an AIC for the random effects model

We will have started with a likelihood for a model at least as general as full time variation on all the parameters, say $\mathcal{L}(\underline{S}, \underline{\theta}) = \mathcal{L}(S_1, \ldots, S_k, \theta_1, \ldots, \theta_\ell)$. Under this time-specific model, $\{S_t, \theta_t\}$, we have the MLEs, $\underline{\hat{S}}$ and $\underline{\hat{\theta}}$, and the maximized log-likelihood, $\log \mathcal{L}(\underline{\hat{S}}, \underline{\hat{\theta}})$ based on $K = k + \ell$. Thus (for large sample size, n), AIC for the time-specific model is the (now) familiar $-2\log \mathcal{L}(\underline{\hat{S}}, \underline{\hat{\theta}}) + 2K$.

The dimension of the parameter space to associate with this random effects model is K_{re}, where

$$K_{re} = \mathrm{tr}(\mathbf{G}) + \ell$$

where **G** is the projection matrix mapping $\underline{\hat{S}}$ onto $\underline{\tilde{S}}$ (see above), and ℓ is the number of free parameters not being modeled as random effects. $\mathrm{tr}(\mathbf{G})$ is the matrix trace (i.e., the sum of the diagonal elements of **G**). The $\mathrm{tr}(\mathbf{G})$ (and thus K_{re}) is generally not integer.*

Note that the mapping of $\underline{\tilde{S}} = \mathbf{G}\underline{\hat{S}}$ is a type of generalized smoothing. It is known that the effective number of parameters to associate with such smoothing is the trace of the smoother matrix.

Finally, the large-sample AIC for the random effects model is

$$\mathrm{AIC} = -2\log \mathcal{L}(\underline{\tilde{S}}, \underline{\tilde{\theta}}) + 2K_{re}$$

* Which is why the number of parameters reported for random effects models is generally not integer.

A more exact version, AIC_c, for the random effects model may, by analogy, be taken as

$$\mathrm{AIC}_c = -2\log\mathcal{L}(\underline{\tilde{S}}, \underline{\tilde{\theta}}) + 2K_{re} + 2\left(\frac{K_{re}(K_{re}+1)}{n+K_{re}-1}\right)$$

For a full derivation of the AIC, in both a fixed and random effects context, see Burnham & Anderson (2002).

D.4. random effects models – some worked examples

In the following, we introduce the 'mechanics' of fitting random effects models in **MARK**, using a series of progressively more complex examples. Many of the steps were introduced earlier in the context of variance components analysis (section D.2). We begin by revisiting the simple binomial ('known fate') analysis we introduced in section D.2.1.

D.4.1. binomial survival revisited – basic mechanics

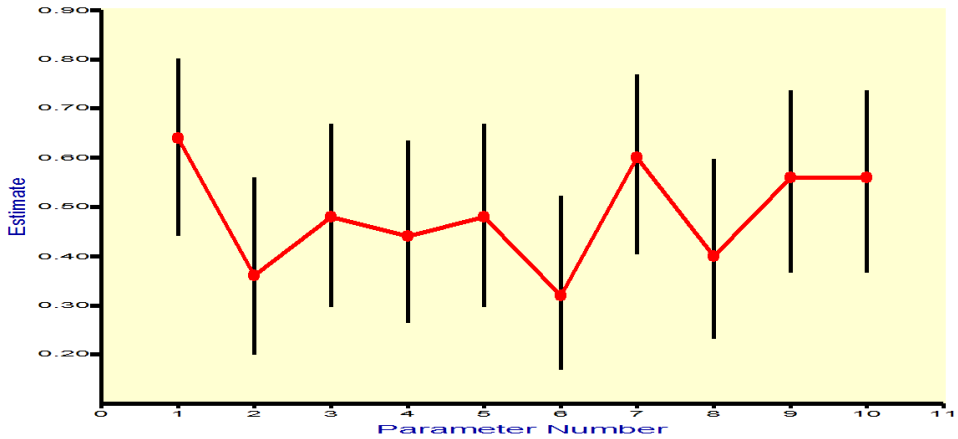
We begin our analysis of the 'known fate' data by considering a candidate set of 2 approximating models: model $\{S_t\}$, and model $\{S_\cdot\}$. Recall that the former model is our general 'time-dependent' model – this is the model we used in our estimation of variance components as detailed in section D.2.1. The second model, $\{S_\cdot\}$ is a model where survival S is constrained to be constant over time. Fit both models,and add the results to the browser:

Results Browser: Known Fate

Model	AICc	Delta AICc	AICc Weight	No. Par.	Deviance
{S(.)}	348.3337	0.0000	0.98724	1	10.2069
{S(t) -- general model}	357.0311	8.6974	0.01276	10	0.0000

Clearly, there is overwhelming support in the data for the time-constant model, $\{S_\cdot\}$.

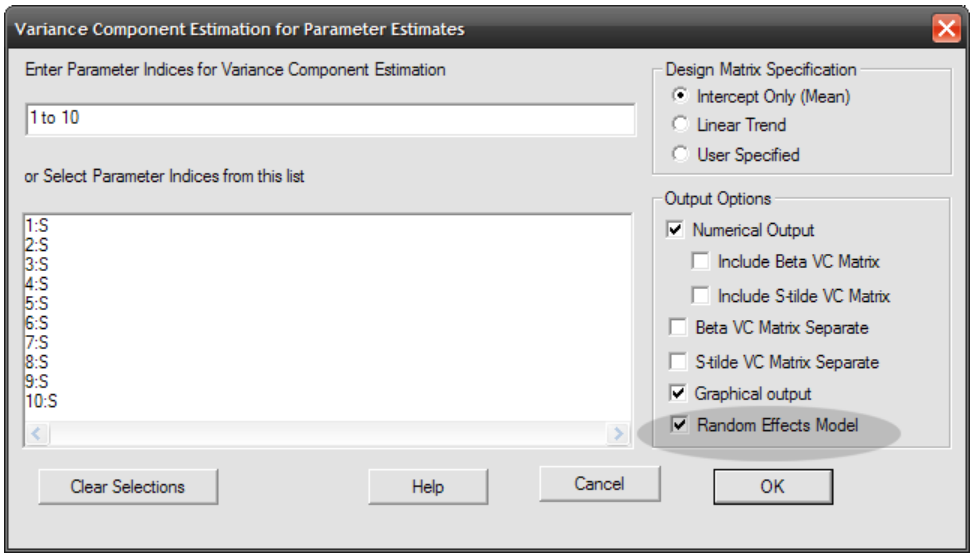
Let's have a look at a plot of the estimates from model $\{S_t\}$:



The point estimates $\hat{S}_i$ appear to vary over time, but notice that the confidence bounds on each estimate appear to overlap over all intervals. This is perhaps the underlying explanation for why, despite apparent variation in the point estimates $\hat{S}_i$, there is essentially no support in the data for the general time-varying model, $\{S_t\}$, relative to a model which constrains the estimates to be constant over time, $\{S_\cdot\}$.

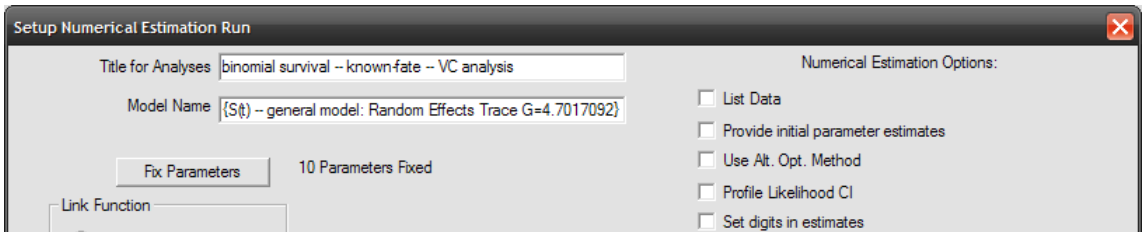
However, we know logically that S_i cannot truly be constant over time. There must be *some* true variation in survival, but our data are insufficient (apparently) to support a time-varying model where '`time`' is modeled as an unconstrained fixed effect. Rather than concluding our analysis of these data here (especially when such a conclusion is based on data insufficiency, rather than biological plausibility), we continue by fitting a random effects model, which we will propose as 'intermediate' between constant models, and fully time-dependent models. We will submit at this point that a random effects model is, in fact, such an intermediate model (support for this statement will be developed in a later section). Here, we will try to fit a model where survival varies around some unknown mean μ, with unknown process variance σ^2 – clearly, this corresponds to the '`intercept only (mean)`' model we first saw in section D.2.1 when we introduced variance components estimation in **MARK**. Such a model also makes some intuitive sense, if our intuition is guided by the time-series plot of the estimate $\hat{S}_i$ shown on the previous page, where it might be reasonable to 'imagine' each S_i as 'bouncing randomly' around some mean survival probability, μ (with the magnitude of the 'bouncing' around the mean being determined by the process variance, σ^2).

Formally, we will refer to this model as $\{S_{\mu,\sigma^2}\}$. How do we build such a model in **MARK**? Easy enough – we simply run through the steps we need for a variance components analysis for model $\{S_t\}$ (which should be familiar to you by now, if you've reached this point in this appendix – if not, go back and review section D.2.1): retrieve our general model $\{S_t\}$, specify parameter indices '`1 to 10`' in the variance component estimation window, and now, before hitting the '**`OK`**' button, we 'check' the box to build the random effects model in the lower right-hand corner of the '**`Variance Component Estimation`**' window:

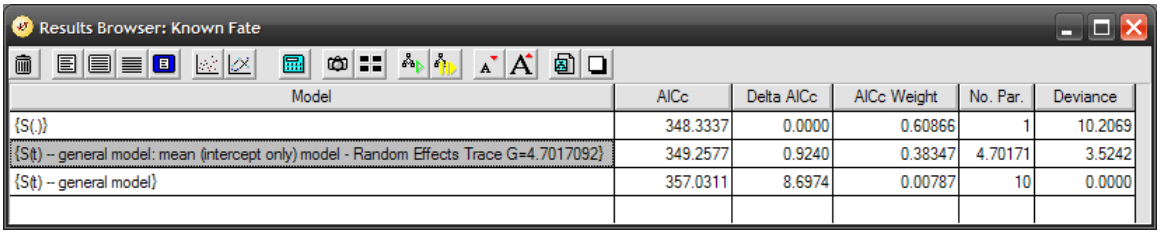


Now, click the '**`OK`**' button. As before, **MARK** will respond by outputting various parameters estimates ($\hat{S}_i$, $\tilde{S}_i$, $\hat{\sigma}^2$) to the editor, and generating a plot of the time-series for the estimates of survival, and the mean model. The derivation and analysis of both was presented in some detail at the beginning of section D.3 and in section D.3.1, so we won't repeat them here.

Here, we introduce the next step, which is the actual 'fitting' of the estimated random effects model to the data. You may have noticed that in addition to the editor window (containing the parameter estimates) and the plot, **MARK** has also brought up a '**`Setup Numerical Estimation Run`**' window, the general contents of which will be familiar to you from other analysis you've done with **MARK**.



The biggest 'visual' difference is that **MARK** has modified the model name. Now, the model is called '`{S(t)f(t) -- sin link: Random Effects Trace G=4.7017092}`'. The part of the model name to the left of the colon is what we originally used to name the model. The part to the right (which **MARK** has added) indicates that we're now running a random effects model, and that the 'trace' of the **G** matrix is 4.7017092. Recall from section D.3.2 that the trace of the **G** matrix is a related to the number of estimated parameters used in the derivation of the AIC (and that because tr(**G**) is generally non-integer, that the number of estimated parameters for random effects models is also usually non-integer). We'll modify the title by adding the words '`intercept only (mean)`' somewhere in the title box, to indicate that the model we're fitting is the '**`intercept only (mean)`**' model. Once done, click the '**`OK to run`**' button and add the results to the browser (if you get a warning about **MARK** not being able to import the variance-covariance matrix, ignore it).



Model	AICc	Delta AICc	AICc Weight	No. Par.	Deviance
{S(.)}	348.3337	0.0000	0.60866	1	10.2069
{S(t) -- general model: mean (intercept only) model - Random Effects Trace G=4.7017092}	349.2577	0.9240	0.38347	4.70171	3.5242
{S(t) -- general model}	357.0311	8.6974	0.00787	10	0.0000

Several things to note here. First, our random effects model now has some significant support in the data (AIC_c weight is 0.383). While not the most parsimonious model in the candidate set, it is clearly better supported than the fixed effect time-dependent model. However, given that a time-invariant model is not logically plausible, then we should select a model where survival varies over time. If we make such a logical choice, then (based on the ideas present in section D.3) our best estimate for annual survival S_i would be the shrinkage estimates $\tilde{S}_i$ from our random effects model, $\{S_{\mu,\sigma^2}\}$.

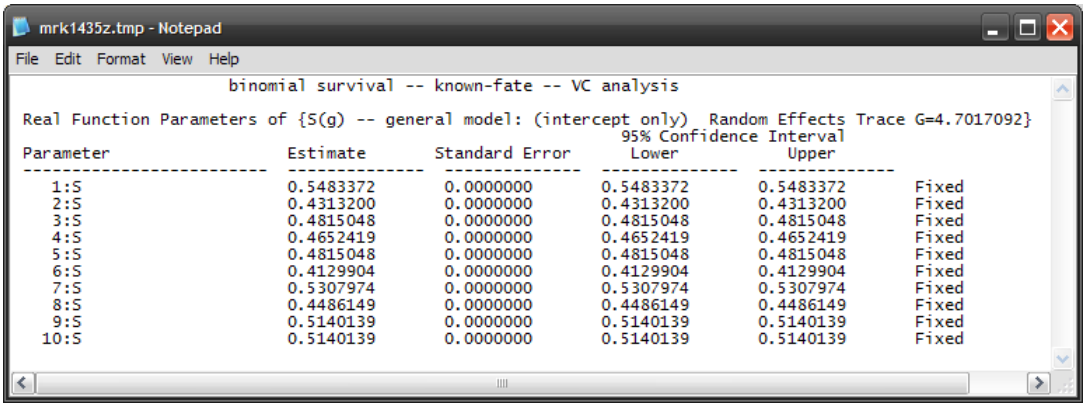
Second, look at the number of parameters that **MARK** reports as having been estimated for this model (4.70171). We see that this number is identical to tr(**G**). Recall from section D.3.3 that the dimension of the parameter space (analogous to the number of estimated parameters in the usual sense) to associate with a random effects model is K_{re},

$$K_{re} = \mathrm{tr}(\mathbf{G}) + \ell$$

where tr(**G**) is the trace of the **G** matrix, and ℓ is the number of free parameters not being modeled as a random effect. In this case, all 10 parameters in the model, $S_1, \ldots, S_{10}$ are being modeled as a random

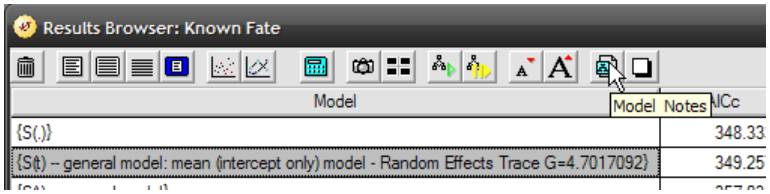
effect, so $\ell = 0$, and thus $K_{re} = \text{tr}(\mathbf{G}) = 4.70171$.

If we next look at the estimates of S_i (below)

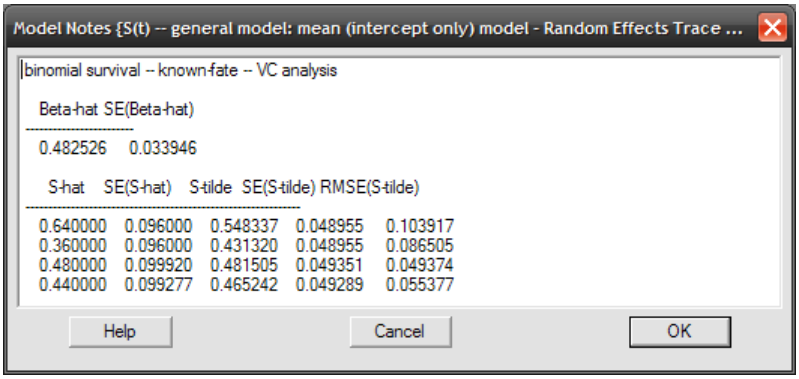


we see that all of the estimates are 'fixed' at the value of $\tilde{S}_i$. As fixed parameters in the fitted model, there is no standard error (or CI) estimated (since there is nothing to be estimated for a fixed parameter, obviously).

Finally, if you click on the '**`model notes`**' button in the browser toolbar



you will be presented with a 'copy' of the variance components analysis which was first output to the editor.



This is convenient, since it allows you to 'store' the variance components analysis for any particular random effects model you fit to the data (note that the variance components analysis is output to the 'model notes' only if you run the random effects model).

D.4.2. a more complex example – California mallard recovery data

Here we introduce the mechanics, and some of the challenges, of fitting of 'random effects' models in **MARK**. We will use a long-term dead recovery data set based on late summer banding of adult male mallards (*Anas platyrhynchos*), banded in California every year from 1955 to 1996 ($k = 41$) (Franklin *et al.*, 2002). The total number of birds banded (marked and released alive) was 42,015, with a total of 7,647 dead recoveries. In a preliminary analysis, the variance inflation factor was estimated as $\hat{c} = 1.1952$. The recovery data are contained in `california-male-mallard.inp`. For your convenience, we've also generated 3 candidate models: $\{S_t f_t\}$, $\{S_T f_t\}$, and $\{S_. f_t\}$, where the capital 'T' subscript is used to indicate linear trend. These models are contained in the associated `.dbf` and `.fpt` files.

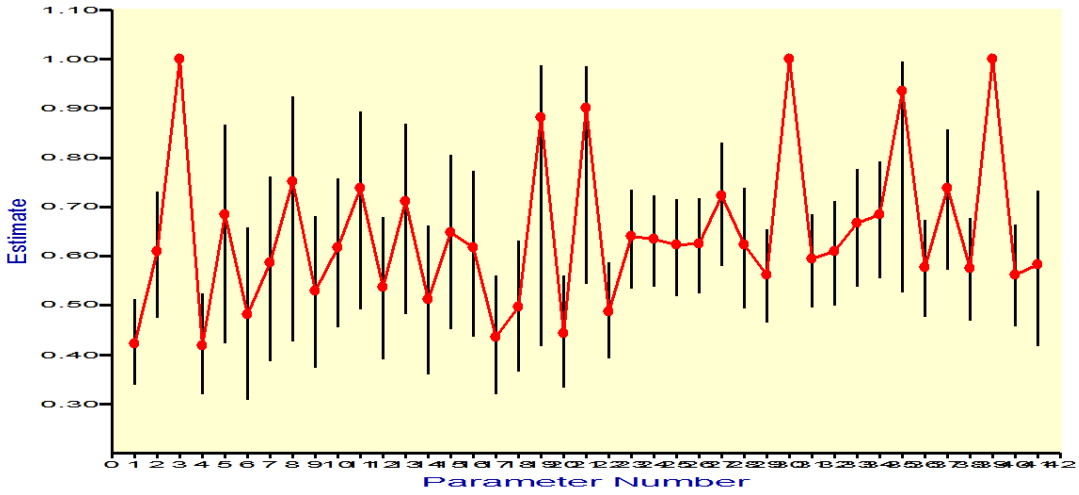
Note that we use time-structure for the recovery parameter f. We do so not simply because such a model often makes more 'biological sense' than a model where f is constrained (say, $f_.$), but because any constraint applied to f will impart (or 'transfer') more of the variation in the data to the survival parameter S, such that the estimated process variance $\hat{\sigma}^2$ will be 'inflated', relative to the true process variance. In general, you want to estimate variance components using a fully time-dependent model, for all parameters, even if such a model is not the most parsimonious given the data.

Here are the results of fitting these 3 models to the data:

Results Browser: Dead Recoveries (Brownie et al.) c-hat = 1.1952000

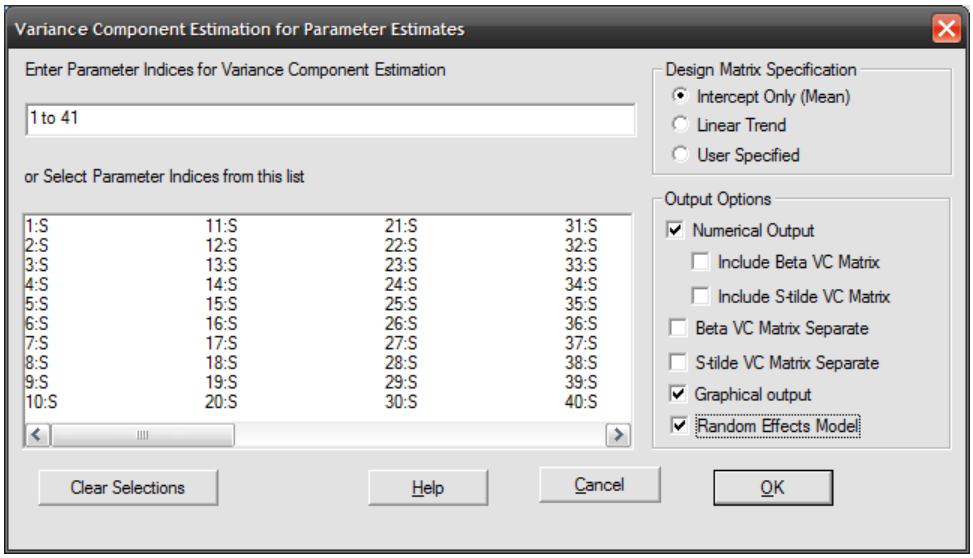
Model	QAICc	Delta QAICc	QAICc Weight	Model Likelihood	No. Par.	QDeviance
{S(t)f(t) -- sin link}	54716.5936	0.0000	0.98132	1.0000	83	368.4517
{S(T)f(t) -- logit link}	54724.5163	7.9227	0.01868	0.0190	44	454.6161
{S(.)f(t) -- sin link}	54803.8298	87.2362	0.00000	0.0000	43	535.9342

Based on the relative degree of support in the data it would seem that there is essentially no support for a model where survival is constrained to follow a linear trend, or for a model where survival is constrained to be constant over time. All of the support in the data (among these 3 models) is for model $\{S_t f_t\}$. If this was all we did, we'd come to the relatively uninteresting and uninformative conclusion that there is temporal variation in survival. A plot of the estimates from this model seems to be consistent with this conclusion:

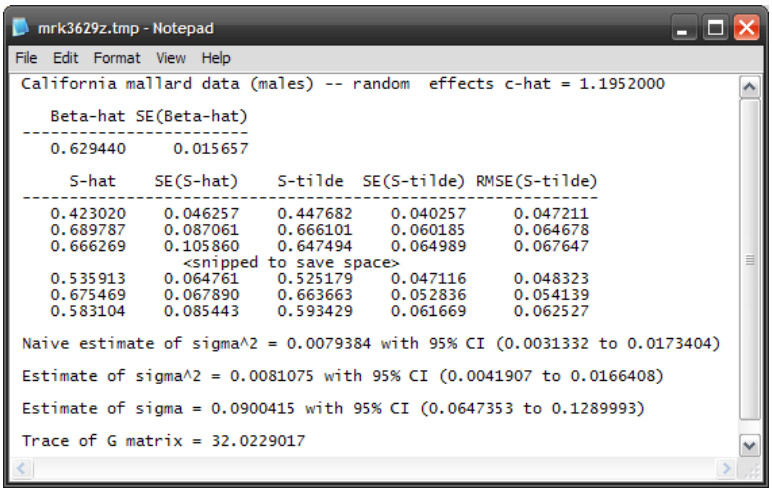


However, rather than concluding our analysis of these data here, or perhaps add some models where annual variation is modeled using a fixed effects approach where annual estimates are constrained to be linear functions of one or more covariates, we might consider models which are 'intermediate' between constant models, and fully time-dependent models. We will submit that a random effects model is, in fact, such an intermediate model.

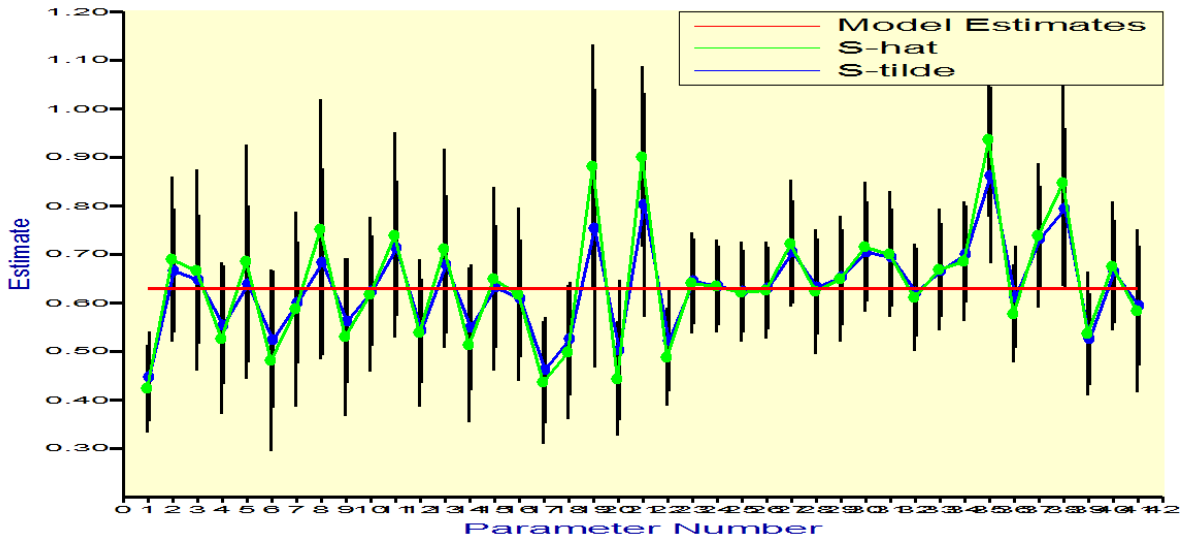
Let's try to fit a model where survival varies around some unknown mean μ, with unknown process variance σ^2 – clearly, this corresponds to the '**`intercept only (mean)`**' model. As was the case in our first example involving binomial 'known fate' survival, we will refer to this model as $\{S_{\mu,\sigma^2} f_t\}$. Go ahead and set up this model, first making sure that the fully time-dependent model is the 'active' model (by retrieving it). Specify parameter indices '`1 to 41`' for the variance component estimation, make sure '**`intercept only (mean)`**' is selected, and that the '**`random effects model`**' button is checked:



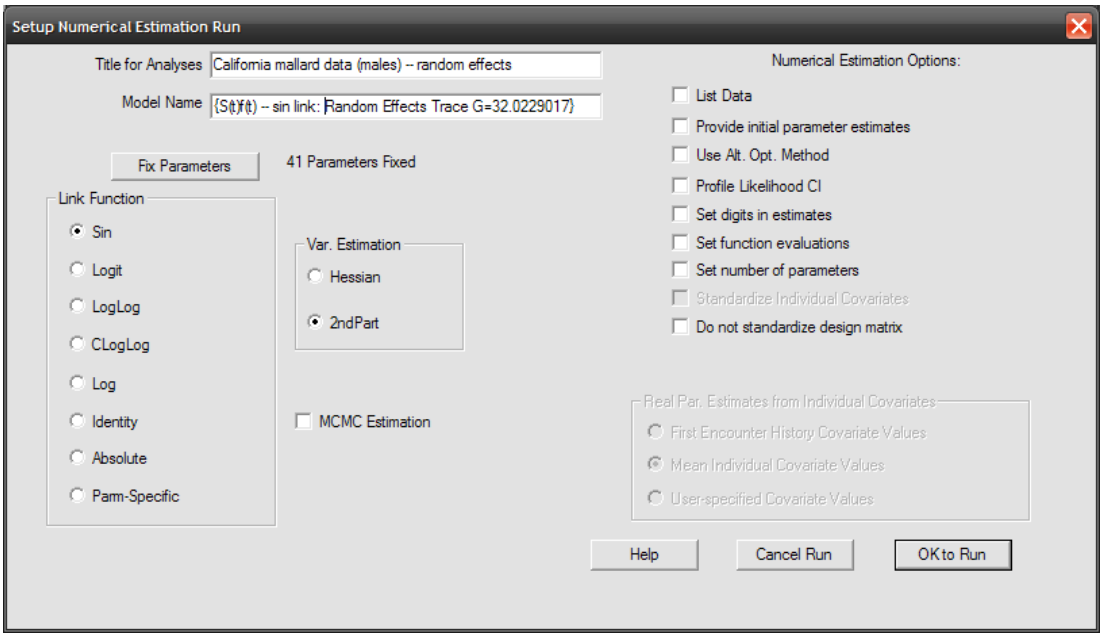
When you click the '**`OK`**' button, you'll be presented with the estimates of the mean, the ML and 'shrinkage' estimates, and the various estimates of the process variance (to save some space, we've snipped out a number of the estimates for S_i).



A plot of the ML and shrinkage estimates, and the model from which the shrinkage estimates were derived (in this case, the intercept only mean model), is shown below:

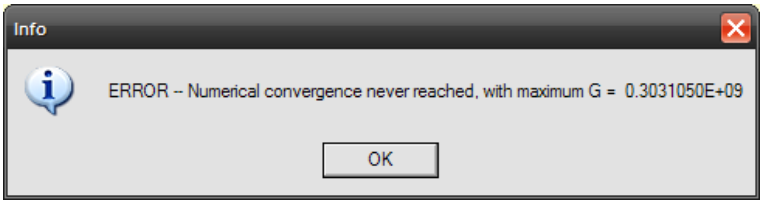


Finally, we come to the estimation run window.



Again, we notice that **MARK** has modified the model name. Now, the model is called'`{S(t)f(t) -- sin link: Random Effects Trace G=32.0229017}`'. The part of the model name to the left of the colon is what we originally used to name the model. The part to the right (which **MARK** has added) indicates that we're now running a random effects model, and that the 'trace' of the **G** matrix is 32.0229. Again, we're going to modify the title slightly, to indicate that the model we're going to fit is the '**`intercept only (mean)`**' model – we'll simply add the words '`intercept only`' somewhere in the title box.

We hit the '**Ok to run**' button and...



Clearly, something has gone wrong. Generally when you see the phrase 'numerical convergence never reached' (or something to that effect) embedded in an error message, your first response should be to consider trying 'better' starting values. Often, such convergence issues reflect some underlying 'problems' with the data (sparseness, one or more parameters estimated near either 0 or 1), and **MARK** is potentially having difficulty estimating the likelihood – a problem which might be exacerbated (or simply an artifact) of the default starting values used in the numerical optimization.

One straightforward approach is to use different starting values – in this case, the ML estimates from model $\{S_t f_t\}$. To do this, simply check the '**provide initial parameter estimates**' box in the numerical estimation run window, before running the random effects model. Now, when you click '**OK to run**', you will be presented with a window asking you to specify the initial parameter estimates for the numerical estimation. To use the estimates from model $\{S_t f_t\}$, simply click the '**retrieve**' button, and select the appropriate model (labeled '`S(t)f(t) -- sin link`'). This will populate the boxes in the '**initial values**' windows with the ML estimates. Then, once you click the '**OK**' button, **MARK** will attempt the numerical optimization. For this example, using these different starting values solves the problem – the random effects model converges successfully.

An alternative approach which also generally works (albeit at the expense of some extended computational time in many cases), and which does not require good starting values for the optimization (which you may not always have), is to use simulated annealing for the numerical optimization. You may recall (from Chapter 10) that you can specify using simulated annealing for the optimization by selecting the '**alternate optimization**' checkbox on the right-hand side of the '**run numerical estimation**' window. What simulated annealing does during the optimization is to periodically make a random jump to a new parameter value. It is this characteristic is what allows the algorithm more flexibility in finding the global maximum (in cases where there may in fact by local maxima in the likelihood; see Chapter 10 for a discussion of this in the context of multi-state models), and minimizes the chances that the numerical solution is determined by starting values (using simulated annealing starts with the defaults, but then makes the random jumps around the parameter space, as described).

To use simulated annealing for our mallard analysis, you simply retrieve model $\{S_t f_t\}$ (our general model), run through the variance components analysis (remembering to check the '**random effects model**' box), and then try again – this time, before hitting the '**OK to run**' button for the generated random effects model, make sure the '**Use Alt. Opt. Method**' button is checked. Change the title (we'll add '**intercept only model -- SA**' to indicate both the model, and the optimization method used to maximize the likelihood), then click '**OK to run**'. Simulated annealing takes significantly longer to converge than does the default optimization routine – how much longer will depend on how fast your computer is. Nonetheless, this approach also works fine, and yields the same model fit as the model fit using different initial values for the optimization. We'll only keep one of these in the browser (shown at the top of the next page).

Results Browser: Dead Recoveries (Brownie et al.) c-hat = 1.1952000

Model	QAICc	Delta QAICc	QAICc Weight	No. Par.	QDeviance
{S(t)f(t) -- sin link: intercept only -- Random Effects Trace G=32.0229017}	54704.8902	0.0000	0.99708	74.02290	374.7711
{S(t)f(t) -- sin link}	54716.5936	11.7034	0.00287	83	368.4517
{S(T)f(t) -- logit}	54724.5163	19.6261	0.00005	44	454.6161
{S(.)f(t) -- sin link}	54803.8298	98.9396	0.00000	43	535.9342

We see that the 'intercept only' random effects model has virtually all the support in the data, even relative to our previous 'best model' $\{S_t f_t\}$. Again, we note that the number of parameters estimated for the random effects model is non-integer. Note, however, that the number of parameters estimated (74.02363) is not simply tr(**G**) (=32.02363). The difference between the two values is (74.02363-32.02363)=42. Where does the 42 come from? Recall that the number of parameters estimated for the random effects model, K_{re} is given as $K_{re} = \text{tr}(\mathbf{G}) + \ell$, where ℓ is the number of free parameters not being modeled as a random effect. In our mallard example, we modeled the 41 survival parameters $S_1, \ldots, S_{41}$ as a random effect, but we left the recovery parameter f modeled over time as a simple fixed effect. How many f parameters in our model? $\ell = 42$, which of course is why the number of parameters estimated is 42 more than tr(**G**).

What more can we about our results so far? Consider the improvement in precision achieved by the shrinkage estimates, $\tilde{S}_i$, from model $\{S_{\mu,\sigma} f_t\}$ compared to the ML estimates, $\hat{S}_i$ from model $\{S_t f_t\}$. As discussed in section D.3.2, a convenient basis for this comparison is the ratio of average $\widehat{RMSE}$ to $\widehat{SE}$:

$$\frac{\widehat{RMSE}(\tilde{S}_i \mid S_i)}{\widehat{SE}(\hat{S}_i \mid S_i)} = \frac{0.06476}{0.07870} = 0.823$$

The average precision of the shrinkage estimates is improved, relative to MLEs, by 18%, hence confidence intervals on S_i would be on average 18% shorter.

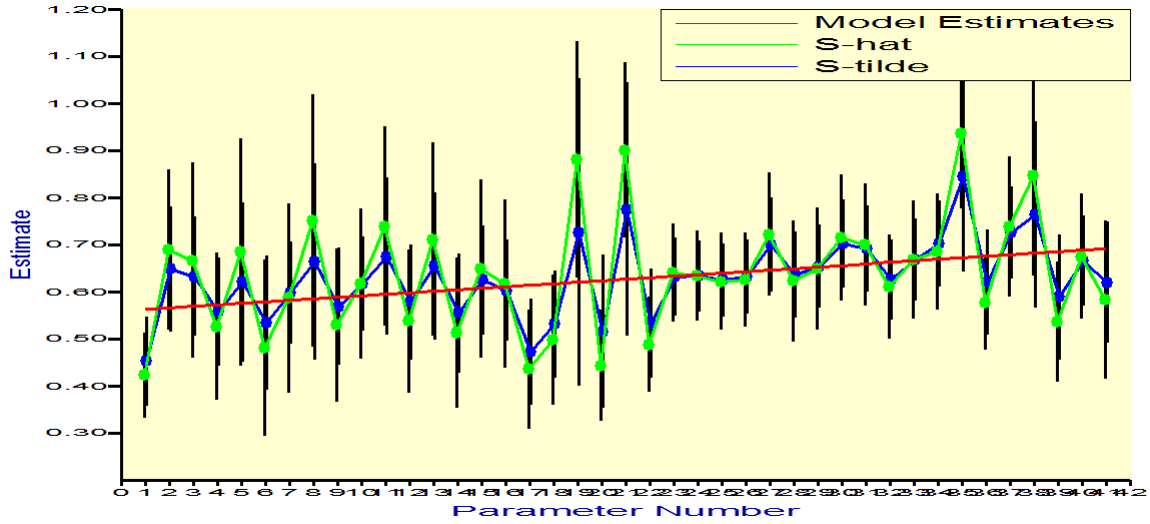
Let's continue by fitting a linear trend random effects model. First, retrieve model $\{S_t f_t\}$. Then, start a '**`variance components`**' analysis – this time, selecting the '**`linear trend`**' design matrix specification, instead of the default '**`intercept only (mean)`**'. Here is a truncated listing of the numerical estimates.

mrk6929z.tmp - Notepad

File Edit Format View Help

```
California mallard data (males) -- random  effects c-hat = 1.1952000

   Beta-hat SE(Beta-hat)
------------------------
   0.559172     0.029216
   0.003350     0.001227

     S-hat    SE(S-hat)    S-tilde  SE(S-tilde) RMSE(S-tilde)
-----------------------------------------------------------------
   0.423019     0.046257    0.447929    0.039253     0.046489
   0.689787     0.087060    0.653587    0.056848     0.067395
   0.666269     0.105858    0.636927    0.060592     0.067323
   0.526082     0.079839    0.554909    0.052654     0.060029
              <snipped to save space>
   0.535913     0.064759    0.542105    0.045904     0.046320
   0.675469     0.067890    0.677862    0.051343     0.051398
   0.583104     0.085443    0.620473    0.059772     0.070492

Naive estimate of sigma^2 = 0.0079390 with 95% CI (0.0031338 to 0.0173410)

Estimate of sigma^2 = 0.0063781 with 95% CI (0.0028298 to 0.0144269)

Estimate of sigma = 0.0798627 with 95% CI (0.0531956 to 0.1201119)

Trace of G matrix = 30.7992796
```

We see that the estimated process variance is nearly half the value estimated from the intercept only model. We also see that the estimate for the slope is positive ($\hat{\beta} = 0.0034$). This is reflected in the plot of the ML and shrinkage estimates against the model, shown below:



Next, we'll go ahead and fit the estimated '**`linear trend`**' RE model to the data, after adding the phrase 'linear trend' to the title.

Here is the results browser with the '**`linear trend`**' random effects model results added:

Results Browser: Dead Recoveries (Brownie et al.) c-hat = 1.1952000

Model	QAICc	Delta QAICc	QAICc Weight	No. Par.	QDeviance
{S(t)f(t) -- sin link: linear trend -- Random Effects Trace G=30.7983785}	54703.3469	0.0000	0.68324	72.79838	375.6855
{S(t)f(t) -- sin link: Random Effects Trace G=32.0229017}	54704.8902	1.5433	0.31583	74.02290	374.7711
{S(t)f(t) -- sin link}	54716.5936	13.2467	0.00091	83	368.4517
{S(T)f(t) -- logit}	54724.5163	21.1694	0.00002	44	454.6161
{S(.)f(t) -- sin link}	54803.8298	100.4829	0.00000	43	535.9342

We see clear evidence of strong support for random variation in the individual S_i around the trend line – this model has almost twice the support in the data as the next best model (our intercept only model). What is of particular note is that if we hadn't built the random effects models, and had based our inference solely on the 3 starting models, we would have concluded there was no evidence whatsoever of a trend, when in fact, the random effects trend model ended up being the best supported by the data.

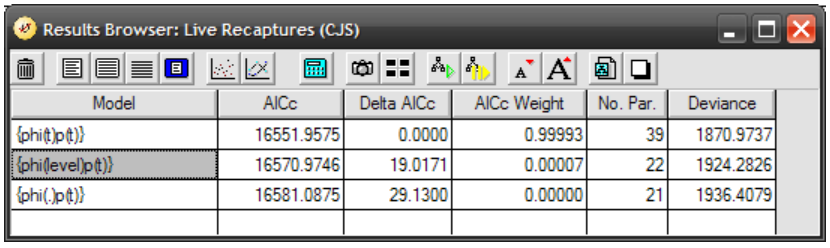
The simple random effects models we used here are both necessary for inference about process variation, σ^2, and also for improved inferences about time-varying survival rates.

D.4.3. random effects – environmental covariates

Here, we consider fitting a random effects model when survival differs as a function of some environmental covariate. Suppose we have some live encounter (CJS) data collected on a fish population studied in a river that is subject to differences in water level. You hypothesize that annual fish survival is influenced by variation in water level. We have $k = 21$ occasions of mark-recapture data (contained

in `level-covar.inp`). Over each of the 20 intervals between occasions, water flow was characterized as either 'average' (`A`) or 'low' (`L`) (more specific covariate information was not available). Here is the time series of flow covariates: {`AAAALLLAAALALALLLLAL`}.

We begin our analysis by considering 3 fixed effect models for apparent survival, φ: $\{\varphi_t p_t\}$, $\{\varphi_. p_t\}$ and $\{\varphi_{level} p_t\}$. Here are the results from fitting these 3 models to the data:

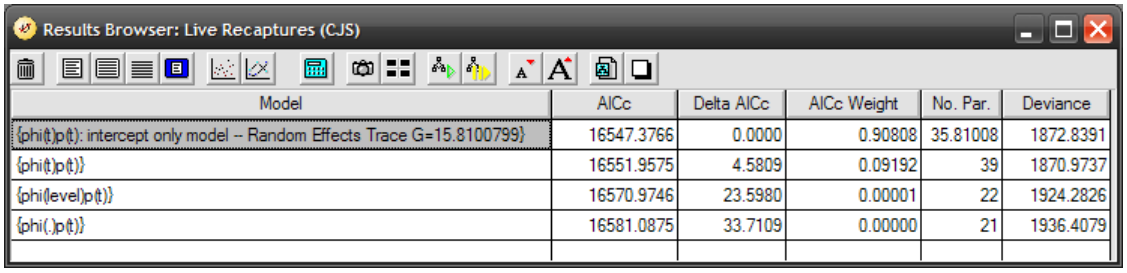

Results Browser: Live Recaptures (CJS)

Model	AICc	Delta AICc	AICc Weight	No. Par.	Deviance
{phi(t)p(t)}	16551.9575	0.0000	0.99993	39	1870.9737
{phi(level)p(t)}	16570.9746	19.0171	0.00007	22	1924.2826
{phi(.)p(t)}	16581.0875	29.1300	0.00000	21	1936.4079

We see strong evidence for variation over time in apparent survival, but no support for an effect of water level. If you look at the estimates from model $\{\varphi_{level}\}$ for average ($\hat{\varphi}_{avg} = 0.709$, SE = 0.0106) and low ($\hat{\varphi}_{low} = 0.650$, SE = 0.0100), the lack of any support for this model may not be surprising. At least, based on considering water level as a fixed effect.

Now let's consider some random effects models. We'll build 2 different models – one a simple intercept only (mean) model, which would seem to be consistent with the strong support for the simple time variation model $\{\varphi_t\}$, and one where survival is thought to vary randomly around a level-specific mean. In other words, we hypothesize $\mu_{low} \neq \mu_{avg}$. We'll assume, however, that $\sigma^2_{low} = \sigma^2_{high}$. This is not directly testable using the moments-based variance components approach in **MARK**, but is testable using an MCMC approach (see appendix E).

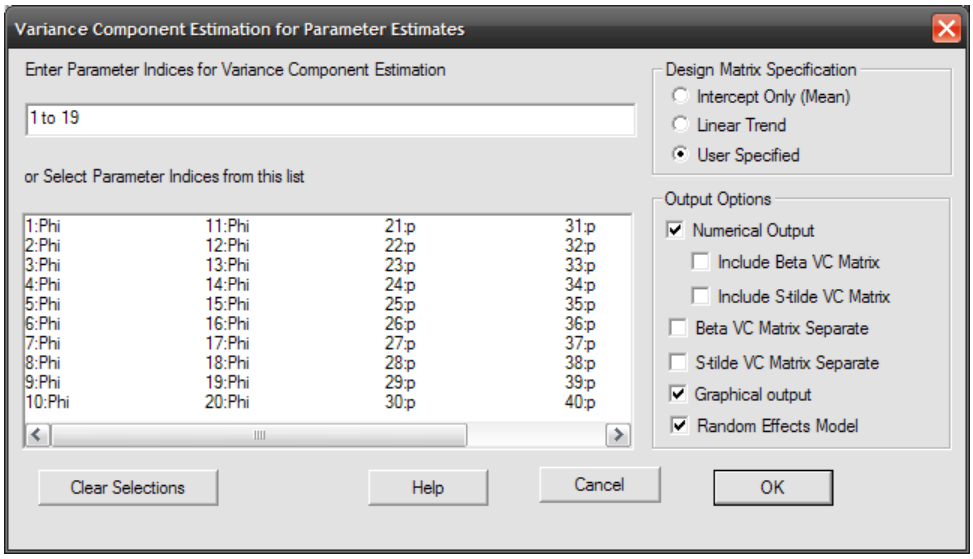
By this point, building and fitting the intercept only model for the survival parameters $\varphi_1 \rightarrow \varphi_{19}$ (remember, we don't include φ_{20} since it is confounded with our estimate of p_{21} for our time-dependent model) should be straightforward, so we'll skip the description of the mechanics, and will simply present the results – we've added the 'intercept only' model to the browser (below).


Results Browser: Live Recaptures (CJS)

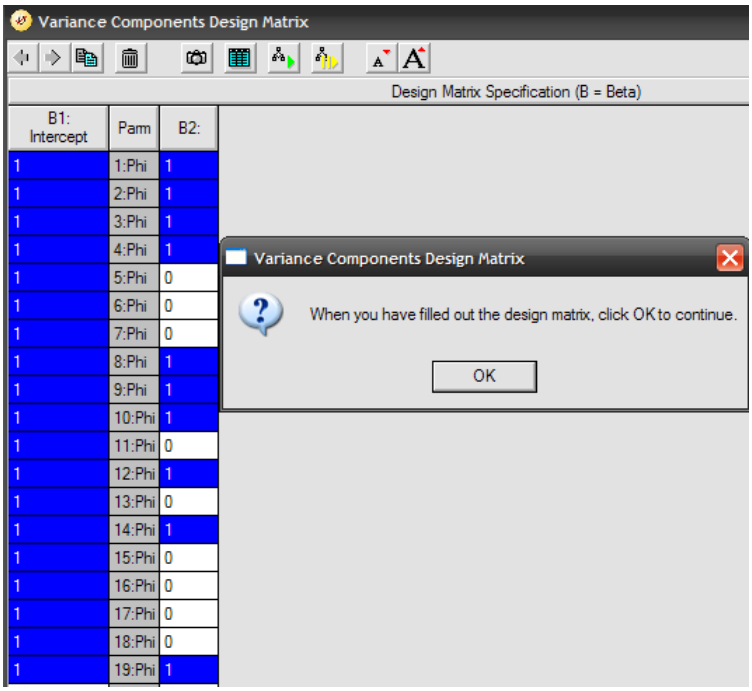
Model	AICc	Delta AICc	AICc Weight	No. Par.	Deviance
{phi(t)p(t): intercept only model -- Random Effects Trace G=15.8100799}	16547.3766	0.0000	0.90808	35.81008	1872.8391
{phi(t)p(t)}	16551.9575	4.5809	0.09192	39	1870.9737
{phi(level)p(t)}	16570.9746	23.5980	0.00001	22	1924.2826
{phi(.)p(t)}	16581.0875	33.7109	0.00000	21	1936.4079

We see strong evidence that the intercept only random effects model is more parsimonious given the data than any other model.

Now, we consider the final model where survival is though to vary randomly around a level-specific mean. We'll refer to this as model $\{\varphi_{\mu_{level}\sigma^2_{level}}\}$. How do we construct this model in **MARK**? Here we finally make use of the '**`User Specified`**' design matrix option in the variance components setup window (shown at the top of the next page). Simply start the variance components analysis, specify the parameters (1 to 20), and check the '**`User Specified`**' design matrix option. Make sure you've also checked the '**`Random Effects Model`**' option as well.



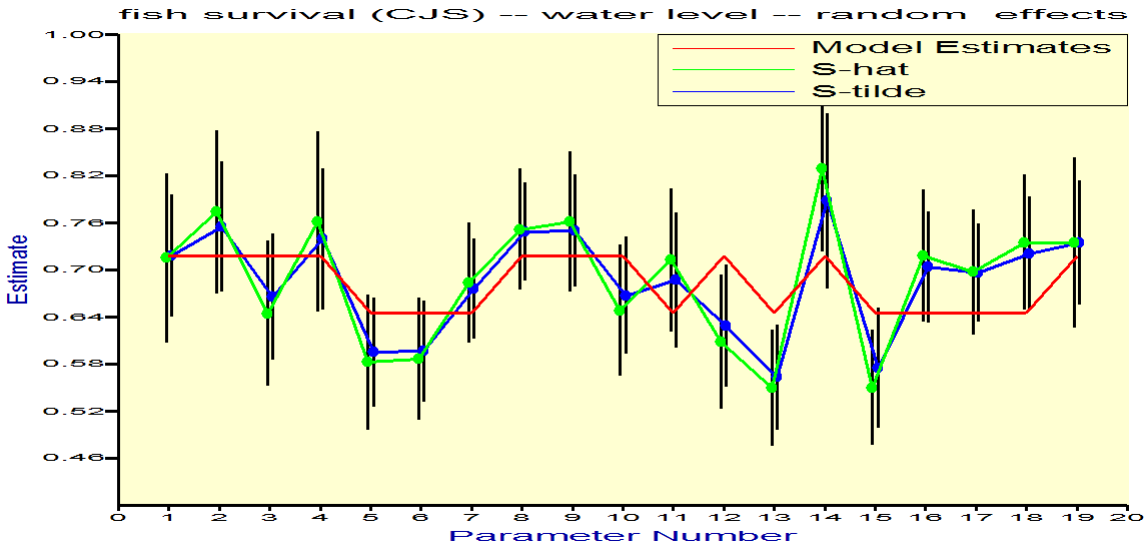
Now, all we need to do is click the '**OK**' button. Once you do so, **MARK** will present you with a small pop-up window asking you to specify how many covariate columns you want in the user-specified design matrix. The default is 3, but here, for our model, we need only 2: one to code for the intercept, and one to code for the water level (a single column since there are only two levels of the water covariate). The design matrix entry window is shown below.



Note that there are only 19 rows in the DM, since we're applying a random effect to $\varphi_1 \rightarrow \varphi_{19}$ only. Also note the small 'pop-up' window indicating that '**When you have filled out the design matrix, click OK to continue**'. Meaning you should do as it says – specify your design matrix for the survival

parameters, and then when you're sure it's correct, click the '**OK**' button.

First we're presented with a plot (below), showing the ML and shrinkage estimates, and (importantly here) the underlying model (the red line).



The red line clearly indicates that there are 2 separate means being modeled, for the low and average water flow years, respectively. The estimated process variance is $\hat{\sigma}^2 = 0.00313$, and the estimate for $\hat{\beta}_1 = 0.0717$ in the linear model indicates that survival is higher in 'average' water level years (since we used 'low' level years as the reference level in our design matrix, above). What is also very important here, is that the shrinkage estimates are clearly not constrained to fall exactly 'on the red line' – they represent shrunk estimates of apparent survival as if each estimate was drawn randomly from a sample with a water level-specific mean.

OK, what about the results of fitting this random effects model to the data?

Results Browser: Live Recaptures (CJS)

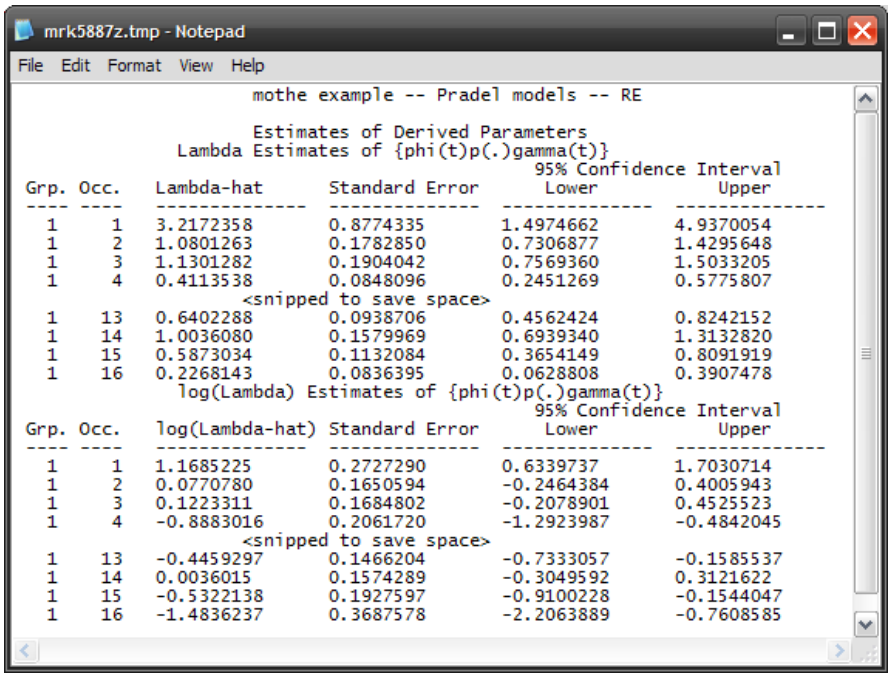
Model	AICc	Delta AICc	AICc Weight	No. Par.	Deviance
{phi(t)p(t): water level model -- Random Effects Trace G=15.2401619}	16546.4694	0.0000	0.58836	35.24016	1873.0840
{phi(t)p(t): intercept only model -- Random Effects Trace G=15.8100799}	16547.3766	0.9072	0.37380	35.81008	1872.8391
{phi(t)p(t)}	16551.9575	5.4881	0.03784	39	1870.9737
{phi(level)p(t)}	16570.9746	24.5052	0.00000	22	1924.2826
{phi(.)p(t)}	16581.0875	34.6181	0.00000	21	1936.4079

What is especially noteworthy here is that the random effects model with water level-specific means, $\{\varphi_{\mu_{level}\sigma^2_{level}}\}$, is now the most parsimonious model in the model set, despite having no support whatsoever when considered in a fixed effects design. The near equivalence of the AIC_c weights between this model and the simpler 'intercept only' random effects model suggest that we can't differentiate between the two, but whereas our initial conclusion strongly rejected the hypothesis that there was an influence of water level on apparent survival, our random effects modeling would suggest that perhaps we shouldn't be quite so sure.

D.4.4. worked example – λ – Pradel model

We conclude with analysis of a famous set of data, the moth (*Gonodontis bidenta*) data reported on by Bishop *et al*. (1978) and compulsively analyzed by many others (e.g., Link & Barker 2005). The data consist of records for 689 male moths that were captured, marked, and released daily over 17 days in northwest England. These moths were nonmelanic; demographic parameters were estimated as part of a larger study looking at comparative fitness of distinct color morphs.*

Here we will use random effect Pradel models (Chapter 13), focussing on estimation of process variance, and possible trend, in realized growth rate λ. The data we'll work with are contained in `moth-example.inp`. Our focus here is on variation in λ. Recall that in a Pradel model, λ an be estimated as either a structural (real) parameter (for data type '**`survival and lambda`**'), or as a derived parameter (for any of the other Pradel model data types). For the purposes of estimating process variance on λ, it doesn't particularly matter which data type we use, since we can estimate process variance for either real or derived parameters. We have already seen variance components analysis of real parameters in earlier examples, so here, we'll demonstrate the process of estimation for λ as a derived parameter. So, let's select the data type '**`Pradel survival and seniority`**'. 17 occasions. Now, you may recall that there are some challenges with parameter confounding for fully time-dependent Pradel models. As is often the case, some of these problems can be handled by applying constraints to one or more parameters. Since our purpose here is simply to demonstrate some mechanics, and not conduct an exhaustive analysis of these data, we'll avoid some of these issues by simply setting the encounter probability p to be constant over time. So, our general model will be $\{\varphi_t p_. \gamma_t\}$. Go ahead and modify the PIM chart to construct this model, and add the results to the browser. Model deviance is 236.708. Here are the derived estimates



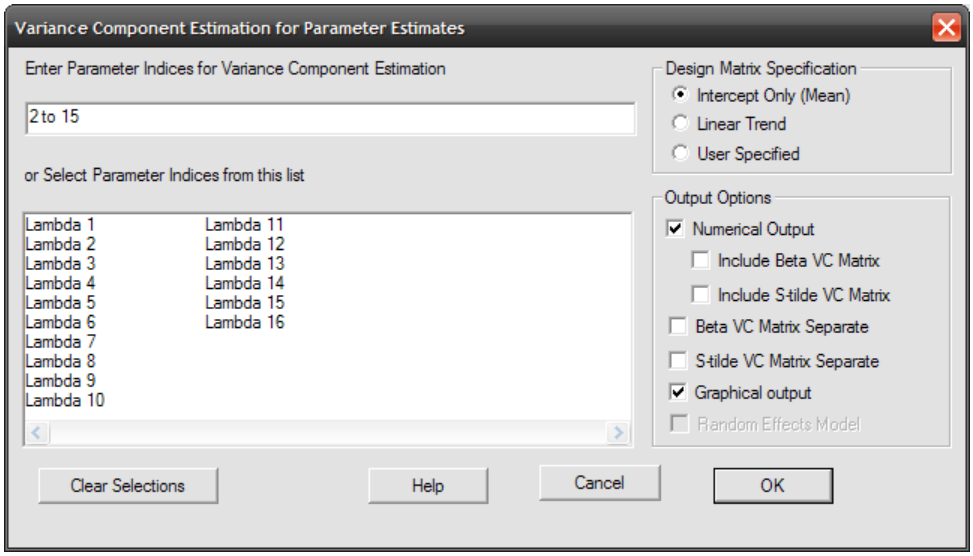
Note that **MARK** generates derived estimates of λ, and log λ. While there are important considerations as to which is more appropriate for analysis, again, our purpose here is more to demonstrate some of the 'mechanics', so we'll focus on estimates of λ on the linear scale.

* The underlying motivation for this study should be very familiar to any of you with some background in evolutionary biology.

Let's construct a random effects model for λ, using $\{\varphi_t p_. \gamma_t\}$ as our general model. The steps are the same, except that when you access the '**variance components**' sub-menu, you need to specify '`derived parameter estimates`' for the parameter type. You'll be asked to select either '**Lambda Population Change**' or '**log(Lambda) Population Change**'. We'll select the former.

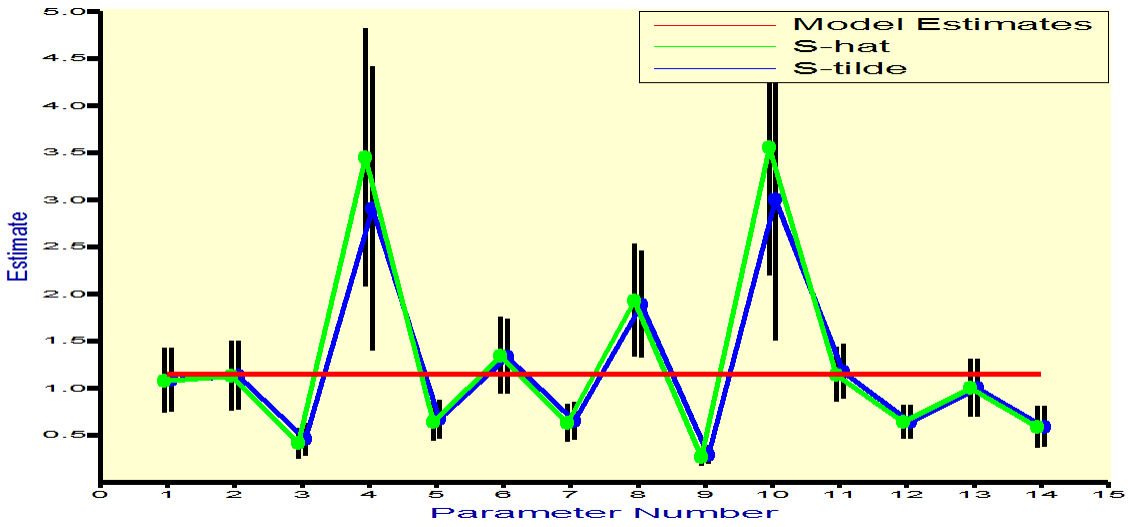
Now, for the first 'challenge' – specifying the parameter index values for λ. You might simply assume that you would select all of them: 1 to 16. However, here (and generally), we need to be a bit careful. When specifying the parameters to include in the random effect, you should not include parameters that are confounded, or that are otherwise known to have 'issues'. For Pradel models, there is potential for confounding, but constraining encounter probability p to be constant over time should solve some of that problem. However, experience has shown that the first and last estimates of λ in Pradel models are often biased. If you look at the estimates for $\hat{\lambda}_1$ and $\hat{\lambda}_{16}$ shown on the previous page, you'll see some evidence that this might be the case for these data. For example, $\hat{\lambda}_1 = 3.217$. This is often a judgement call (for example, you might say that $\hat{\lambda}_{11} = 3.55$ is even larger, so using scale as evidence for a 'problem' with $\hat{\lambda}_1$ is perhaps not a great criterion). For now, we'll be 'conservative', and include only $\lambda_2 \rightarrow \lambda_{15}$ in specifying the random effect.

Now, for one important difference in fitting random effects models to derived parameters – you can't. If you look in the lower-right-hand corner, you'll notice that the '**Random Effects Model**' check-box has been 'greyed-out'.



If you think about it, this should make some sense. The random effects 'constraint' can be applied only to real (structural) parameters in the model. Here, we are estimating λ as a derived parameter (i.e., by algebra), and thus we can't build and fit a random effects model using the Pradel '**Survival and seniority only**' data type. We'll revisit this point in a moment. Go ahead and click the '**OK**' button. The plot of the real estimates $\hat{\lambda}_i$ and shrinkage estimates $\tilde{\lambda}_i$ is shown at the top of the next page. The estimated mean $\hat{\bar{\lambda}} = 1.147$, with an estimated process variance of $\hat{\sigma}^2 = 0.722$. The plot shows clear evidence of fairly large swings in realized growth – this is not overly surprising for an insect, where large changes in reproduction and survival are relatively commonplace (there aren't enough data available to test for 'boom-bust' cycles, another common occurrence with insect growth dynamics).

Now, what if we wanted to fit a random effects model for λ, and not simply estimate the mean and process variance? To do this, we need to change the data type, to one where λ is included as a



structural parameter in the model. Simply select '**PIM | Change data type**'. You will be presented with a box contained the different data types you can apply to these data. You'll see that one of those data types presented is '**Pradel survival and lambda**'. Since that is the only model containing λ as a real structural parameter, select that data type, and click '**OK**'.

You'll find that you've been returned back to the browser view – with no indication that anything has changed. But, if you look at the PIM chart, you'll see that the current 'active' model structure is $\{\varphi_t p_t \lambda_t\}$. In other words, the parameter structure of the model has changed (as expected), but the underlying model has reverted back to the fully time-dependent model. **MARK** has basically assumed that if you're switching data types, you are (in effect) starting over. So, we want to make the encounter probability p constant, so that our general model under this data type, $\{\varphi_t p_. \lambda_t\}$ is consistent with the general model we used under the data type where λ was estimated as a derived parameter.

Go ahead and run model $\{\varphi_t p_. \lambda_t\}$, and add the results to the browser.

Results Browser: Pradel Survival and Lambda

Model	AICc	Delta AICc	AICc Weight	No. Par.	Deviance
{phi(t)p(.)lambda(t)}	5062.9674	0.0000	1.00000	32	190.6262
{phi(t)p(.)gamma(t) -- general model}	5111.2040	48.2366	0.00000	33	236.7079

Unfortunately we see that these are not the same models, even though in theory they should be – the model deviances and the number of estimated parameters are both different. There might be a couple of issues here. First, if you look at the estimates, you'll see that the first and final estimates of apparent survival φ and the final estimate of λ are both poorly estimated, and are probably confounded (despite setting p constant). Second, and perhaps more likely, the logical constraint that $\lambda_i > \varphi_i$ is not enforced for 1 or more parameters. This is not uncommon in fitting the '**Pradel survival and lambda**' model (for discussion of this issue, see the - `sidebar` - beginning on p. 9 of Chapter 13). As such, comparing model $\{\varphi_t p_. \lambda_t\}$ with $\{\varphi_t p_. \gamma_t\}$ makes little sense, although the estimates of $\hat{\bar{\lambda}} = 1.170$ and $\hat{\sigma}^2 = 0.697$ from model $\{\varphi_t p_. \lambda_t\}$ (using $\lambda_2 \rightarrow \lambda_{15}$) are fairly close to the estimates derived for model $\{\varphi_t p_. \gamma_t\}$.

For the moment, we'll skip over this issue, and focus on some additional 'mechanics'. Delete model $\{\varphi_t p_. \gamma_t\}$ from the browser – we'll focus on model $\{\varphi_t p_. \lambda_t\}$. We are interested in analysis of trend in λ. Trend in realized λ is potentially of great significance for conservation and management, and robust estimation of trend would seem to be something worth pursing. Now, based on the plot of $\hat{\lambda}_i$ and $\tilde{\lambda}_i$ shown at the top of the preceding page, we don't expect a 'trend' model of any flavor to have much support in the data. But, to demonstrate the mechanics, we'll proceed anyway.

With model $\{\varphi_t p_. \lambda_t\}$, we can see there are at least 2 ways we could proceed. We could build an ultrastructural model where λ is constrained to fall on a straight trend line, or we could build a random effects trend model. We'll do both, for purposes of comparison. To make the comparison 'fair', we'll build the 'T' (trend) ultrastructural model applied to $\lambda_2 \rightarrow \lambda_{15}$ only, since these are the parameters we would use in the random effects model. Then, we'll build the random effects 'trend' model, also using $\lambda_2 \rightarrow \lambda_{15}$ only, and add the results of both to the browser.

Results Browser: Pradel Survival and Lambda

Model	AICc	Delta AICc	AICc Weight	No. Par.	Deviance
{phi(t)p(.)lambda(t)}	5062.9674	0.0000	0.52787	32	190.6262
{phi(t)p(.)lambda(t): Random Effects Trace G=13.4052579}	5063.1906	0.2232	0.47213	31.40526	192.1287
{phi(t)p(.)lambda(T)} -- lam(2)-lam(15) only}	5167.1277	104.1603	0.00000	18	324.4423

We notice immediately that the random effects model with trend actually gets a fair amount of support in the data, especially relative to the ultrastructural fixed effects trend model. So, despite appearances, there might be some evidence of trend in the data. Enough that the overall support for this model is fairly compelling.

At least 2 points to make here. First, there are some general concerns with the ultrastructure approach, where constraints are applied to λ. Since growth is a function of *per capita* survival plus *per capita* recruitment, $\lambda = \varphi + f$, then any constraint applied to λ enforces a strict negative covariance between $\hat{\varphi}$ and $\hat{f}$ (see chapter 12, and discussions in Franklin 2001). Such a covariance is clearly artificial, and may make little to no biological sense (since it would imply that any increase in survival is perfectly balanced by an equal and opposite change in recruitment). Second, the random effects approach to trend analysis can't be applied to λ when λ is estimated as a *derived* parameter – you can estimate the intercept and slope of the trend, and the process variance, but you can't fit a random effects model for a derived parameter (i.e., you cannot add it to the browser).

While there are some technical challenges with Pradel models in general (especially for time-dependent models, which are exacerbated for random effects models which are generally applied to time-dependent models), it seems clear that considering variance components and random effects analysis for λ in Pradel models has potential to be extremely useful.

D.5. Model averaging?

At this point, you might be asking yourself, 'what about model averaging?'. This important topic is introduced in some detail in Chapter 4 (see the Burnham & Anderson 'model selection' book for a comprehensive treatment).

There are at least two issues to consider here. First, if our candidate model set contains both fixed effect and random effect models, should we model average over the entire set? In fact, **MARK** does

nothing 'mechanically' to prevent you from doing so. For example, take the mallard analysis introduced in section D.4.2. Assume that we have the following 5 models in the browser – 3 fixed effects models ($\{S_t f_t\}$, $\{S_. f_t\}$, $\{S_T f_t\}$), and 2 random effect models ('`linear trend model`', and '`intercept only (mean) model`':

Results Browser: Dead Recoveries (Brownie et al.) c-hat = 1.1953000

Model	QAICc	Delta QAICc	QAICc Weight	No. Par.	QDeviance
{S(t)f(t) -- identity link: linear model -- Random Effects Trace G=29.2792764}	54697.9994	0.0000	0.64024	71.27928	377.9199
{S(t)f(t) -- identity link: intercept only (mean) model -- Random Effects Trace G=31.2637772}	54699.1555	1.1561	0.35917	73.26378	375.0929
{S(t)f(t) -- identity link}	54712.0300	14.0306	0.00057	83	368.4209
{S(T)f(t) -- identity link}	54719.3733	21.3739	0.00001	44	454.0064
{S(.)f(t) -- identity link}	54799.2522	101.2528	0.00000	43	535.8894

You can go ahead and run the model averaging routines on the survival estimates – **MARK** simply assumes you want to average over the entire model set. Your only hint that you might need to be a bit careful comes when you look at the model averaging output. Here is a snippet of the output for the mallard analysis, for the parameter corresponding to S_1:

mrk8148z.tmp - Notepad

File Edit Format View Help

```
            California mallard data (males) -- random effects
        Estimates only for data type Dead Recoveries (Brownie et al.)
                          c-hat = 1.1953000

                  Survival Parameter (S) Group 1 Parameter 1
Model                                    Weight    Estimate      Standard Error
---------------------------------------- -------   ------------- --------------
{S(t)f(t) -- identity link: linear model 0.64024   0.4526058     0.0000000      Invalid parameter estimate with zero SE?
{S(t)f(t) -- identity link: intercept on 0.35917   0.4503414     0.0000000      Invalid parameter estimate with zero SE?
{S(t)f(t) -- identity link}              0.00057   0.4230193     0.0462586
{S(T)f(t) -- identity link}              0.00001   0.5634913     0.0102959
---------------------------------------- -------   ------------- --------------
Weighted Average                                   0.4517771     0.0000267
Unconditional SE                                                 0.0017521
95% CI for Wgt. Ave. Est. (logit trans.) is 0.4483453 to 0.4552135
Percent of Variation Attributable to Model Variation is 99.98%
```

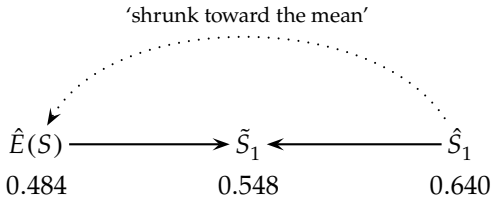
Pay particular attention to the right-hand side. Notice that for the 2 random effects models, you get a 'warning' about 'invalid parameter estimates' with 'zero SE?'. **MARK** has 'noticed' that the parameter estimates for parameters modeled as random effects (the survival parameters, in this case) have a zero SE. **MARK** 'suspects' there might be a problem, but in fact this is exactly what you should see. As discussed earlier, the random effects model is fit to the data after fixing the random parameters to their shrinkage estimates, and thus, there is no SE for those parameters. And, since model averaging intends to generate an unconditional parameter estimate, by accounting for conditional uncertainty in the parameter for each model, then there are clear problems when one or more of the parameters 'appear' to be estimated without any uncertainty (i.e., have zero SE's). So, if **MARK** included the random effect parameter estimates in the model averaging, then the estimate of the unconditional SE for that parameter would be very negatively biased (i.e., much too small).

The second issue is 'conceptual', and relates broadly to the issue of 'model redundancy'. In the extreme, having 2 structurally identical models in the same candidate model set would clearly render model averaged estimates invalid. Model redundancy in the context of AIC selection is discussed in Burnham and Anderson (sections 4.2.9 and 4.2.10). With some thought, you might see that the random effects model is substantially redundant to its fixed effects likelihood version. For example, model $\{S_{\mu,\sigma} f_t\}$ would be redundant to some degree to model $\{S_t f_t\}$.

In one sense, a random effects model is a model which is intermediate between a time-invariant ('dot', say $S_.$) model, and a fully time-varying fixed effects model (say, S_t). Recall from section D.3 that shrinkage estimates $\tilde{S}_i$ generally lie between $\hat{E}(S)$ and $\hat{S}_i$. Recall from section D.3.1 that in the absence of sampling covariance, the shrinkage estimator used in **MARK** is

$$\tilde{S}_i = \hat{E}(S) + \sqrt{\frac{\hat{\sigma}^2}{\hat{\sigma}^2 + \hat{E}_S\left[\mathrm{var}(\hat{S}_i \mid S_i)\right]}} \times \left[\hat{S}_i - \hat{E}(S)\right]$$

Consider the estimates of survival for the first interval from the binomial example presented in section D.2.1: the ML estimate $\hat{S}_1 = 0.640$, the mean survival $\hat{E}(S) = 0.484$, and the shrunk estimate $\tilde{S}_1 = 0.548$. As shown in the following



the ML estimate is 'shrunk' toward the mean, with the resulting shrinkage estimate, $\tilde{S}_1$ being 'intermediate' between the mean, $\hat{E}(S)$ (which is equivalent to a 'dot' model, in some respects) and the ML estimate $\hat{S}_1$ (which comes from the 'time-dependent' model). So, the shrinkage estimate is analogous to an 'average' between the two estimates. As noted in section D.3, the degree of shrinkage is determined by the magnitude of the process variance σ^2, relative to the sampling variance, $\hat{E}_S[\mathrm{var}(\hat{S}_i \mid S_i)]$. If the process variance is relatively small, then the shrunk estimates will approach the estimate for the mean $\hat{E}(S)$, whereas if the process variance is large relative to the sampling variance, then the shrunk estimates will be closer to the ML estimates, $\hat{S}_i$. So, the shrinkage model $\{S_{\mu,\sigma}, \underline{\theta}\}$, is *intermediate* between model $\{S_t, \underline{\theta}\}$ and model $\{S_., \underline{\theta}\}$, and is thus strongly analogous to an 'average model'. As a result, when the process and sampling variances are similar, the estimates from model $\{S_{\mu,\sigma}, \underline{\theta}\}$ will tend to be quite similar to the model-averaged estimates estimated over model $\{S_t, \underline{\theta}\}$ and model $\{S_., \underline{\theta}\}$.

D.6. caveats, warnings, and general recommendations

Using random effects as a basis for modeling collections of related parameters is a long-standing approach in statistics and one that can be very effective. Use of the random effects approach in capture-recapture is relatively new – in the nearly 10 years since the publication of B&W, there have been relatively few applications of these models to real data, despite what we believe are several interesting opportunities made available by these methods.

However, we also believe that the methodology needs to be better understood as to any potential pitfalls and as to its operating characteristics. The following is a summary of our experience to date with random effects models, particularly as implemented in **MARK**. This material is largely abstracted from B&W, and accumulated experience with such models since the time of that publication.

1. the 'method of moments' described in the appendix, and as implemented in **MARK**, has been shown to perform well, especially when $\sigma^2 > 0.025$. The method(s) may not do so well if $\sigma^2 \to 0$. However, we think it reasonable to believe that for a worthwhile study yielding good data, process variation, σ^2, will generally not be too small, relative to average

sampling variation and it is for these conditions (of 'good data') that we need effective random effects inference methods.

2. Another issue to be aware of, as regards estimation of the parameter σ^2, is the matter of unequal, rather than equal length, time intervals. Let the time interval i have length Δ_i. Then we should parameterize the model as $S_i = (\psi_i)^{1/\Delta_i}$ where now each survival probability ψ_i is on the same unit time basis. It may then make biological sense to consider parameters that are a mean and variation for $\psi_1, \ldots, \psi_k$. But this may just as well not make sense, because the time intervals are intrinsically not comparable as they may be in very different times of the annual cycle. It becomes a subject matter judgement as to whether random effects analysis will be meaningful with unequal time intervals. For the moment, don't apply random effects models or variance components analysis to situations where the intervals between sampling occasions are unequal (even specifying unequal interval length will generally yield negatively biased estimates of process variance, σ^2).
3. In practice if there is overdispersion in the data, as measured by a scalar often denoted by c (see Chapter 4 and Chapter 5), the estimated sampling variance-covariance must be adjusted by a reliable $\hat{c}$ (see discussion in B&W, and Franklin *et al*. 2002 for an example with real data).
4. A key design feature to focus on to meet the criterion of 'having good data' when applying random effects is k, the number of estimable random effects parameters (time intervals, locations, etc.). The sample size for estimating σ^2 is k. Therefore, one must not have k too small; < 10 is too small. Even if we knew all the underlying S_i a sample of size $k < 10$ is too small for reliable inference about the variation of these parameters (even given a random sample of them, which is not required here). Inference performance has been shown to be acceptable when $k > 15$. The benefits (includes shrinkage estimates) of random effects models become greater as the number of underlying parameters, k, increases.
5. The other influential design feature is number of individuals marked and released. Both numbers initially released and numbers recaptured are important to the performance of inferences from random effects models. While there are no 'hard and fast' rules, we can make some general recommendation. B&W showed that low numbers of marked and released animals, especially for low survival and encounter probabilities, generally led to point estimates of $\sigma^2 \rightarrow 0$ – this is because the sampling variation was much larger than process variance in these cases.
6. The situation where inferences from a random effects model are most advantageous seems to be for when σ^2 is about the same as average sampling variance, $(\hat{S}_i \mid S_i)$ (recall that sampling variance is strongly influenced by sample size of animals capture and reencountered, whereas process variance is not). If one or the other variance component dominates the total variation in the MLE's $\hat{S}_i$ then the data strongly favor either the simple model $\{S_\cdot p_t\}$ (sample variance dominates), or the general model $\{S_t p_t\}$ (process variation dominates), rather than the random effects model.

   However, it is not a problem, as regards inference about σ^2, to have large sample sizes of animals, hence small sampling variances, so that should be one's design goal. If it then turns out that sampling variance is similar to process variance, the random effects model will be markedly superior to model $\{S_t p_t\}$. Thus, in a sense the random effects model is optimal at the 'intermediate' sample size case. As sample size of animals increases, the random effects model converges to model $\{S_t p_t\}$.
7. A potential technical issue is the 'boundary effect' (at least under what is basically a likelihood approach). As discussed in B&W, if one enforces the constraint $S < 1$ when

the unbounded MLE $\hat{S} \geq 1$, then standard numerical methods used in **MARK** to get the observed information matrix fails. As a result, the estimated information matrix is incorrect for any terms concerning the $\hat{S}$ that is at the bound of 1 (and the inverse information matrix is likely wrong in all elements). Experience shows that, in this case, the resultant point estimate of σ^2 can be very different from what one gets when the survival parameter MLE's are allowed to be unbounded. The difference can be substantial. Using an identity link, B&W found $\hat{\sigma}^2$ to be unbiased in many cases.* With good data we rarely observe an unbounded MLE of S that exceeds 1. This might be explored in a Bayesian context, where it is easy (in a MCMC analysis) to allow S to have its distribution over an interval such as 0 to 2 (rather than 0 to 1). B&W considered this, and found a strong effect of the upper-bound on the point estimate (and entire posterior distribution) for σ^2, and for that particular S. (*Note*: MCMC applications in **MARK** are discussed in Appendix E).

D.7. Summary

This appendix has considered random effects models. The name 'random effects' can be misleading in that a person may think it means that underlying years or areas (when spatial variation is considered, rather than temporal) must be selected at random. This is neither true, nor possible, for a set of contiguous years. Variance components is a better name, in that at the heart of the method is the separation of process and sampling variance components. The issue of what inferential meaning we can ascribe to σ^2 is indeed tied to design and subject matter considerations. However, the shrinkage estimators do not depend on any inferential interpretation of σ^2; rather, they can always be considered as improvements, in a MSE sense, over the MLEs based on full time-varying S_t. The random effects model only requires that the residuals $(S_i - \mathbf{XB})$ are exchangeable.

When are we interested in these sorts of models? Often, if a data set is sparse, but with many (≥ 10) occasions, model $\{S(.)\}$ will be selected. Clearly, this is a *model*, as we know that conditional survival probability cannot remain exactly the same over any significant length of time. While model $\{S(.)\}$ might be 'best' in the sense of a bias-variance trade-off (i.e., it is identified by AIC as most parsimonious among the candidate models), it might leave the investigator wondering about the variation in the parameters. Thus, the estimation of σ^2 has relevance. At the other extreme, assume that model $\{S_t\}$ is selected; here the investigator might have (say) 25 estimates of the survival parameters, each perhaps with substantial sampling variation. This makes it difficult to see patterns (e.g., time trends or associations) or understand the variation in the parameters. Further, analysis of stochastic population models is often complicated by uncertainty concerning the relative variation in estimates of one or more demographic parameters.

Random effects models are a very interesting class of models which can address both issues (amongst many more), but even a partial understanding is somewhat difficult to achieve. The intent of this appendix was to try to convey the general notion of random effects models, and the idea of 'variance components', as implemented in program **MARK**, in a reasonably accessible fashion. The subject of random effects models and variance components as applied to data from marked individuals is treated in considerably more depth in several of the following papers. An alternative approach to estimating variance components, based on Bayesian inference (*sensu* Royle and Link, 2002) and Markov Chain Monte Carlo (MCMC), is presented in Appendix E.

* Note that we do not suggest routinely accepting final inferences that include survival estimates exceeding 1. In fact, the shrinkage estimates will generally not exceed 1, so using $\tilde{S}_i$ and not $\hat{S}_i$ will be the needed improved inference. However, to get to this final inference it may be desirable to pass through an imaginary space ($S > 1$), just as imaginary numbers can facilitate real solutions to real problems. Models only need to possess utility, not full reality.

D.8. Literature

Burnham, K. P., D. R. Anderson, G. C. White, C. Brownie, and K. H. Pollock. 1987. *Design and analysis methods for fish survival experiments based on release-recapture*. American Fisheries Society Monograph No. 5. Bethesda, Maryland, USA. 437 pp.

Burnham, K. P., and G. C. White. 2002. Evaluation of some random effects methodology applicable to bird ringing data. *Journal of Applied Statistics*, **29**, 245-264.

Franklin, A. B., D. R. Anderson, and K. P. Burnham. 2002. Estimation of long-term trends and variation in avian survival probabilities using random effects models. *Journal of Applied Statistics*, **29**, 267-287.

Pfister, C. A. 1998. Patterns of variance in stage-structured populations: evolutionary predictions and ecological implications. *Proceedings of National Academy of Science*, **95**, 213-218.

Royle, J. A., and W. A. Link. 2002. Random effects and shrinkage estimation in capture-recapture models. *Journal of Applied Statistics*, **29**, 329-351.

Schmutz, J. A. 2009. Stochastic variation in avian survival rates: life-history predictions, population consequences, and the potential responses to human perturbations and climate change. Pages 441-461 in D. L. Thomson, E. G. Cooch, and M. J. Conroy, editors. *Modeling Demographic Processes in Marked Populations*. Springer, Berlin.

White, G. C. 2000. Population viability analysis: data requirements and essential analyses. Pages 288-331 in L. Boitani and T. K. Fuller, editors. *Research techniques in animal ecology: controversies and consequences*. Columbia University Press, New York, New York, USA.

White, G. C., K. P. Burnham, and D. R. Anderson. 2001. Advanced features of Program MARK. Pages 368-377 in R. Field, R. J. Warren, H. Okarma, and P. R. Sievert, editors. *Wildlife, Land, and People: Priorities for the 21st Century*. Proceedings of the Second International Wildlife Management Congress. The Wildlife Society, Bethesda, Maryland, USA.

White, G. C. 2000. Population viability analysis: data requirements and essential analyses. Pages 288-331 in L. Boitani and T. K. Fuller, editors. *Research techniques in animal ecology: controversies and consequences*. Columbia University Press, New York, New York, USA.

APPENDIX E

Markov Chain Monte Carlo (MCMC) estimation in MARK . . .

Markov Chain Monte Carlo (more conveniently, MCMC) is a parameter estimation procedure that is frequently (but not exclusively) associated with Bayesian inference, that has been implemented in **MARK** for 2 primary purposes:

1. to provide the capability to more flexibly model and estimate the mean and variance (i.e., variance decomposition) of both univariate and multivariate *hyperdistributions* (i.e., the joint distribution of 2 sets of parameters)
2. to provide the capability to derive more intuitive credibile intervals for the estimated parameters.

In this appendix, we discuss the basic theory and mechanics of using MCMC in **MARK**, for a variety of problems which would be difficult (at best) to implement in any other way. We defer a review of the theory and mechanics underlying MCMC to an Addendum at the end of the appendix. If you have no background at all in MCMC, or Bayesian inference, you are encouraged to have a look at the Addendum before proceeding too far in this appendix.* While it is possible to proceed in applying MCMC in **MARK** without a fair understanding of 'how it works', this is counter to our view that you are always better off if you actually know a bit about 'what **MARK** is doing'. Those of you with stronger backgrounds might want to flip through the Addendum at some stage, if only to give you some insights as to the details of how things are implemented in **MARK**.

It is assumed that the reader already has a basic knowledge of some standard encounter-mark-reencounter models as described in detail in this book (e.g., dead recovery and live recapture models – referred to here generically as capture-recapture). We also assume familiarity with the variance components and random effects models presented in Appendix D – we strongly suggest you work through Appendix D in full before continuing here, if you have not done so already.

We introduce the subject of – and some of the motivation for – this appendix by example. In the following we consider two relatively common scenarios (out of a much larger set of possibilities) where a 'different analytical approach' (i.e., MCMC) might be helpful at least, or essential (in the case of the second example).

* *Note*: we do not mean to imply that this appendix, or the addendum to same, are in any way a substitute for formal study of MCMC in general, and Bayesian inference in particular. What we present here is only intended as a minimum sufficient introduction to get you started. Take a class, or study one of the many very good books on the subject.

© Cooch & White (2014)

scenario 1 – parameters as random samples

Consider a Cormack-Jolly-Seber (CJS) time-specific model $\{S_t p_t\}$ wherein survival (S) and capture probabilities (p) are allowed to be time varying for $(k + 2)$ capture occasions, equally spaced in time. If $k \geq 20$ we are adding many survival parameters into our model as if they were unrelated; however, more parsimonious models are often needed. Consider a reduced parameter model – at the extreme, we have the model $\{S_\cdot p_t\}$ wherein $S_1 = S_2 = \cdots = S_k = S_\cdot$. However, this model may not fit well even if the general (time-dependent) CJS model fits well and there is no evidence of any explainable structural time variation, such as a linear time trend, in this set of survival rates, or variation as a function of an environmental covariate. Instead, there may be unstructured time variation in the S_i that is not easily modeled by any simple smooth parametric form, yet which cannot be wisely ignored. In this case it is both realistic and desirable to conceptualize the actual unknown S_i as varying, over these equal-length time intervals, about a conceptual population mean $E(S) = \mu$, with some population variation, σ^2 (Fig. E.1).

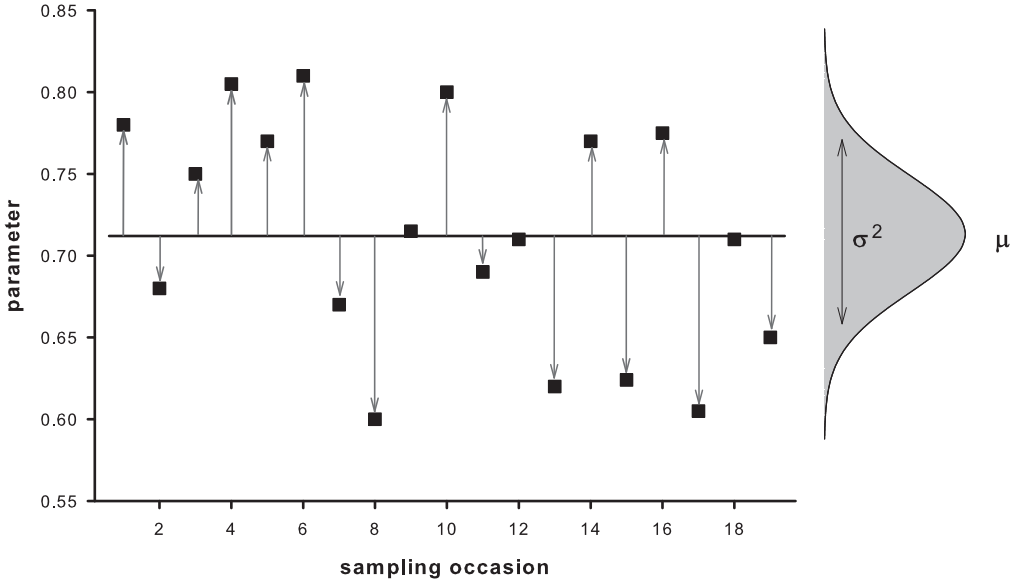


Figure E.1: *Schematic representation of variation in occasion-specific parameters θ_i, as if the parameters were drawn randomly from some underlying distribution with mean μ and variance σ^2.*

Here, by population, we will mean a conceptual statistical distribution of survival probabilities, such that the S_i may be considered as a sample from this distribution. Hence, we proceed as if S_i are a *random* sample from a distribution with mean μ and variance σ^2. The parameter σ^2 is now the conventional measure of the unstructured variation in the S_i, and we can usefully summarize $S_1 \ldots S_k$ by two parameters: μ and σ^2. The complication is that we do not know the S_i; we have only estimates $\hat{S}_i$, subject to non-ignorable sampling variances and covariances, from a capture-recapture model wherein we traditionally consider the S_i as fixed, unrelated parameters. We would like to estimate μ and σ^2, and adjust our estimates to account for the different contributions to the overall variation in our estimates due to sampling, and the environment.

In Appendix D, we considered estimation of these 2 parameters using a random effects model based on a 'methods of moments' approach. We will see in this appendix how we can not only estimate μ and

σ^2 using MCMC, but how MCMC will allow much greater flexibility for more complex problems than the 'method of moments' approach.

scenario 2 – covariation between 2 structural parameters

In Chapter 8, we introduced 'dead recovery' models – so named because the 'encounter data' consist of recoveries of dead marked individuals. One dead recovery parametrization (Brownie) is commonly used for analysis of recovery data where the mortality event is influenced by harvest. For harvested species, an individual marked and released alive can experience one of 3 fates (Fig. E.2): (1) it can survive the year with some probability (S), (2) it can be 'harvested' (i.e., some 'action' leading to permanent removal) with some probability (K), or (3) it can 'die' from 'natural' causes with probability ($1 - S - K$) (i.e., it might actually die from some reason other than harvest, or permanently emigrate the sampling area, at which point it appears dead) . Conditional on being harvested, (i) the individual may be retrieved (probability c), and (ii) reported (i.e., the individual identification number of the harvested individual is submitted to some monitoring agency), with probability λ). The product ($Kc\lambda$) is referred to as the 'recovery rate', f.

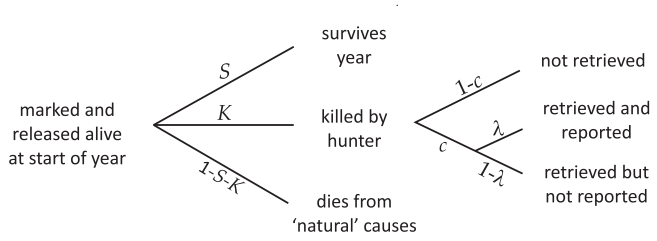
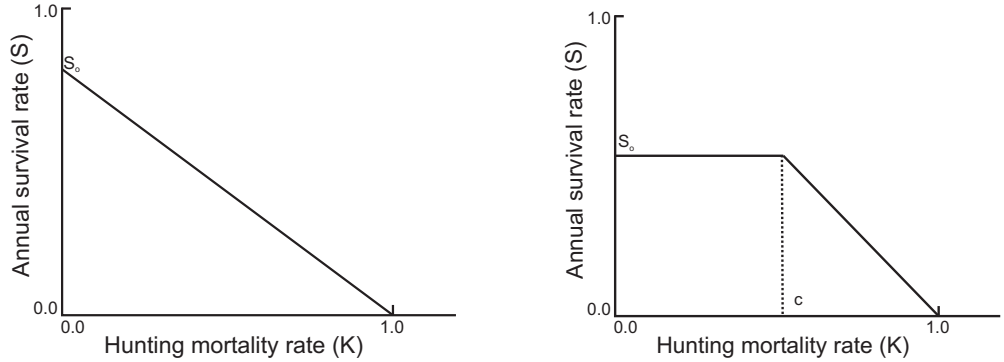


Figure E.2: *Probability of different fates for marked individuals subject to harvest.*

Note that f is related to the 'survival' process, since an individual which is shot, retrieved and reported, does not survive. So, there is some anticipated structure relating S and f. Traditionally, the relationships between survival and harvest are broadly dichotomized as reflecting either *additive* or *compensatory* mortality (Fig. E.3).



(a) additive mortality hypothesis **(b)** compensatory hypothesis

Figure E.3: *Patterns of variation in survival, S, and mortality rate due to harvest, K.*

In the case of additive mortality (Fig. E.3a), the sources of mortality are, just that, 'additive' – they add together, so that both natural and harvest mortality are combined in some additive way. In contrast, with compensatory mortality (Fig. E.3b), we are, in effect, assuming that we harvest only those animals which were likely to have died from natural causes in the first place - harvest does not increase the overall mortality rate (at least over a certain range). Moreover, if survival is a constant over some range, even when mortality due to harvest is increasing, then this implies that over that range, natural mortality is decreasing! This is the key difference between compensatory and additive mortality – in compensatory mortality, natural mortality varies as a function of how much mortality there is due to harvest, while in additive mortality, the 2 simply add together. Mechanistically, compensatory mortality is generally believed to reflect the process(es) of density-dependence, while additive mortality reflects density-independence. This dichotomy between additive and compensatory mortality was first articulated by Anderson & Burnham in 1976.

If we assume that $c\lambda$ is a constant, then any variation in f must be proportional to variation in K. And thus, the structural relationship between S and f potentially yields important insights in differentiating between the additive and compensatory hypotheses, which has important implications for harvest management (since K, and thus f are under management control, at least to some extent; see the Williams, Nichols & Conroy (2002) book for an exhaustive treatment of the subject). If you look at figures (E.3a) and (E.3b) for a moment, it should be fairly clear that negative process correlation between harvest and survival rates is consistent with at least partially additive harvest effect on survival, whereas process correlations > 0 are consistent with a hypothesis that harvest mortality is fully compensated by other sources of mortality (Anderson & Burnham 1976).

In theory, then, all we need to do is look at the correlation of estimates of S and f. Easy enough in principle, but recall that estimates of these two parameters are generally not independent – there is significant sampling covariance within and between parameters. Thus, we can't simply take the estimates and 'do statistics on statistics' (i.e., calculate the bivariate correlation ρ between S and f). Fortunately, MCMC provides a solution, because MCMC approaches produce parameter estimates that are not influenced by sampling covariance. So, we can use the MCMC capabilities in **MARK** to derive a robust estimate of S and f, and the correlation between them.*.

The ability to model the covariance between structural parameters in a statistically valid way using MCMC might also be of some interest for many other data types. For example, a negative correlation between recruitment f and apparent survival φ in Pradel models (Chapter 13) might be consistent with some hypotheses concerning cost of reproduction. Another example might be the relationship between abundance and the probability of temporary emigration in robust design models (Chapter 15), where a negative correlation between N and γ'' (the probability of temporarily emigrating) might be consistent with some hypotheses concerning density-dependence of individuals temporarily leaving the sampled population (which might be of interest if, say, the sampled population represents only breeding individuals, and temporary emigration is equivalent to non-breeding). Still another example might involve looking for interesting relationships between apparent survival S and state transition probabilities ψ, in multi-state models (Chapter 10).

In this appendix, we will consider application of MCMC to estimation of variance components, and modeling the relationship amongst structural parameters, using program **MARK**. We will outline the 'mechanics' of using program **MARK**, by means of a series of 'worked examples'.

* For example, this approach has been applied to greater sage-grouse recovery data – Sedinger, J. S., G. C. White, S. Espinosa, E. R. Parte & C. E. Braun. (2010) Assessing compensatory versus additive harvest mortality: an example using greater sage-grouse. *Journal of Wildlife Management*, **74**, 326-332. For an experimental approach, see Sandercock, B. K., E. B. Nilsen, H. Brøseth & H. C. J. Pedersen. (2011) Is hunting mortality additive or compensatory to natural mortality? Effects of experimental harvest on the survival and cause-specific mortality of willow ptarmigan. *Journal of Animal Ecology*, **80**, 244-58.

E.1. Variance components analysis revisited - MCMC approach

In Appendix D, we introduced the concept of 'process variation' among a set of parameters, and the mechanics of estimating process variation in **MARK**, using a moments-based estimator. Here we introduce an alternative approach, using MCMC.

E.1.1. Example 1 - binomial survival re-visited

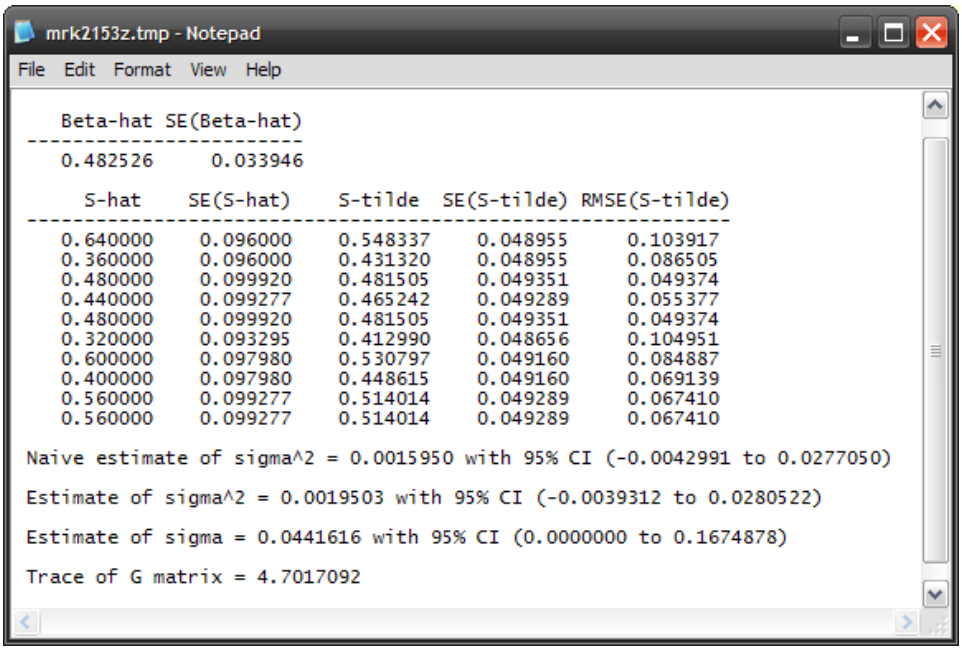
We start by re-visiting the binomial survival example introduced in Appendix D. Again, we imagine a scenario where we are conducting a simple 'known fate' analysis (Chapter 16). In each of 10 years ($k = 10$), we mark and release $n = 25$ individuals, and determine the number alive, y, after 1 year (since this is a known-fate analysis, we assume there is no error in determining whether an animal is 'alive' or 'not alive' on the second sampling occasion). Here, though, we'll assume that the survival probability in each year, S_i, is drawn from $\mathcal{N}(0.5, 0.05)$ (i.e., distributed as an independent normal random variable with mean $\mu = 0.5$ and process variance $\sigma^2 = 0.05^2 = 0.0025$). Conditional on each S_i, we generated y_i (number alive after one year in year i) as an independent binomial random variable $B(n, S_i)$. Thus, our maximum likelihood estimate of survival for each year is $\hat{S}_i = y_i/n$, with a conditional sampling variance of $\widehat{\text{var}}(\hat{S}_i \mid S_i) = [\hat{S}_i(1-\hat{S}_i)]/n$, which given $\mu = 0.5$, and $\sigma^2 = (0.05)^2 = 0.0025$, is approximately 0.01.

Table (E.1) gives the values of S_i, y_i and $\hat{S}_i$ for our 'example data'. Clearly, for a 'real analysis', we would not know the true values for S_i – we would have only $\hat{S}_i$, and generally only have $\hat{E}_S\left(\text{var}(\hat{S}_i \mid S_i)\right)$ as $\widehat{\text{var}}(\hat{S}_i \mid S_i)$. From Table (E.1) we see that the *empirical* standard deviation of the 10 estimated survival rates (i.e., the $\hat{S}_i$) is 0.106. However, we should not take $(0.106)^2$ as an estimate of σ^2 because such an estimate includes *both* process and sampling variation. Clearly, we want to subtract the estimated sampling variance from the total variation to get an estimate of the overall process variation.

Table E.1: *Single realization from simple binomial survival example,* $k = 10$, $E(S) = 0.5$, $\sigma = 0.05$, *where* $\hat{S}_i = y_i/n$ *are* $B(25, S_i)$, *hence expected* $SE(\hat{S}_i|S) \approx 0.1$.

year (i)	S_i	$\hat{S}_i$	$\widehat{\text{SE}}(\hat{S}_i \mid S_i)$
1	0.603	0.640	0.096
2	0.467	0.360	0.096
3	0.553	0.480	0.100
4	0.458	0.440	0.100
5	0.506	0.480	0.100
6	0.498	0.320	0.093
7	0.545	0.600	0.098
8	0.439	0.400	0.098
9	0.488	0.560	0.099
10	0.480	0.560	0.099
mean	0.504	0.484	0.100
SD	0.050	0.106	

In Appendix D, we applied an approach based on a linear 'method of moments'. The results from the variance components analysis of these data presented in Appendix D are shown below:

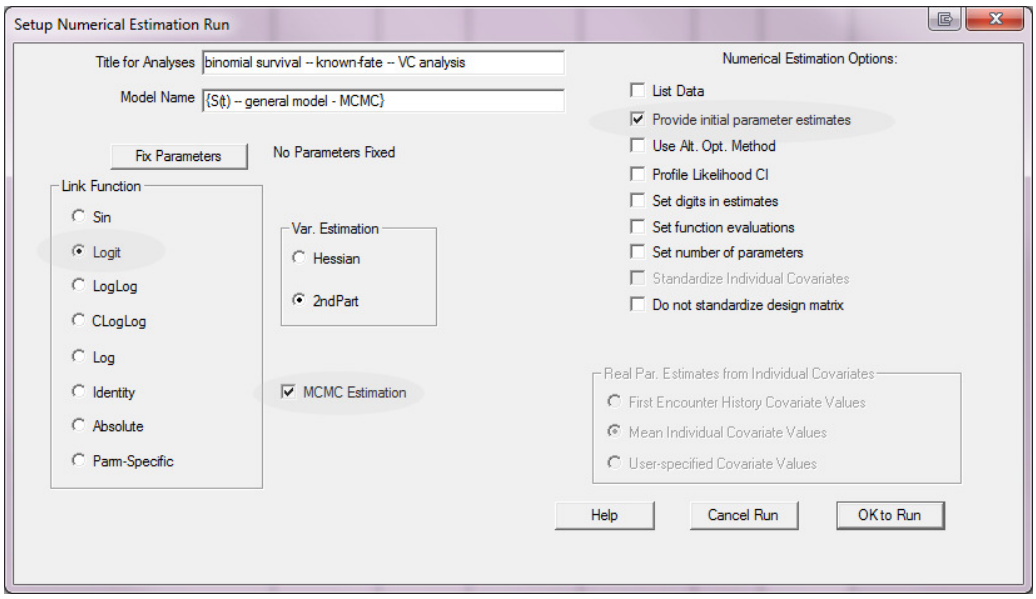


Starting from the top – the first line of the output (above) reports a '`Beta-hat`' of 0.482526. As discussed in Appendix D, this is our most robust estimate of the mean survival probability. This estimate is followed by the estimate of '`SE(Beta-hat)`' which is our most robust estimate of total variance (i.e., process + sampling variation). In the absence of sampling covariance, it is estimated as the square-root of the sum of estimated process variation, $\hat{\sigma}^2$, and sampling variation, $E[\widehat{\text{var}}(\hat{S}_i \mid S_i)]$, divided by k, where k is the number of parameter estimates.

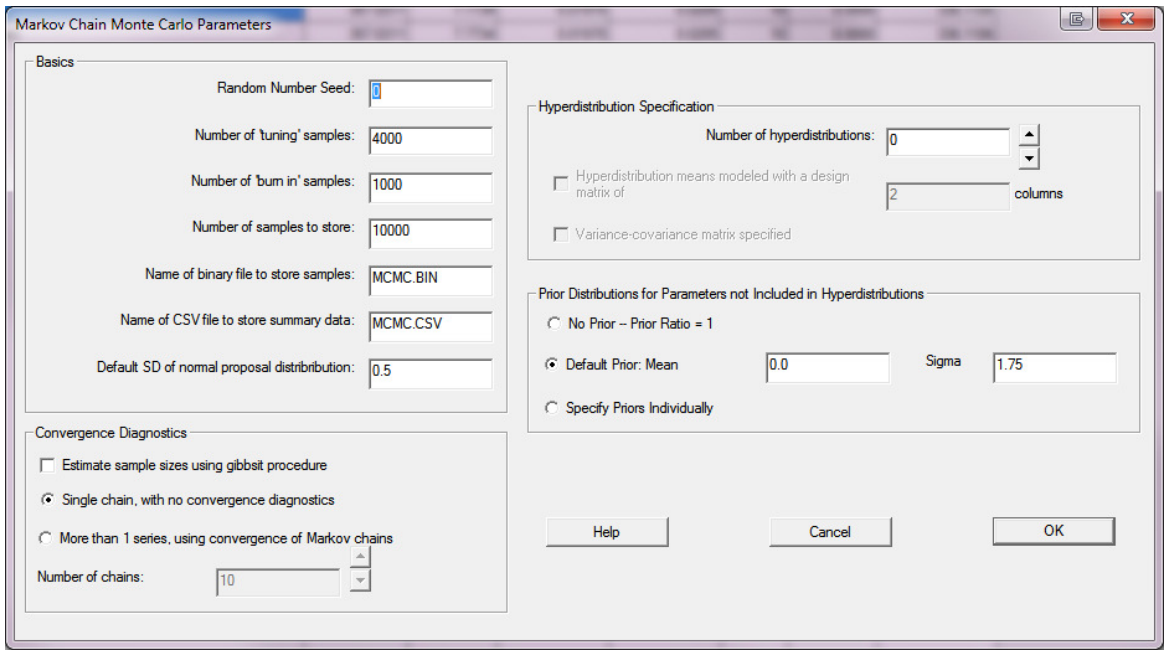
Next, a table of various parameter estimates. The first two columns are the ML estimates of survival $\hat{S}_i$ ('`S-hat`'), followed by the standard error for the estimate ('`SE(S-hat)`'. Next, the 'shrinkage' estimates $\tilde{S}_i$ ('`S-tilde`') and corresponding SE and RMSE.

Finally, the estimates for process variation. First, **MARK** reports the '`Naive estimate of sigma^2`'= 0.001595. This is followed by the '`Estimate of sigma^2`' = 0.0019503 (and the '`Estimate of sigma`' = 0.044162). As discussed in Appendix D, these are the 'preferred' estimates for process variance. We concentrate here on using MCMC in **MARK** to derive similar – hopefully, identical – estimates for σ.

We begin by opening up the `binomial-example.dbf` file (from Appendix D). The browser should contain at least the general model $\{S_t\}$. Retrieve the general model. We're now going to re-run it, but this time, making use of the MCMC capabilities in **MARK**. Click the '**Run**' icon in the toolbar, and bring up the '**Setup Numerical Estimation Run**' window (shown at the top of the next page). Here, notice that we've checked the '**MCMC Estimation**' box, indicating we want to use the MCMC capabilities in **MARK** for the estimation. We've also checked the '**Provide initial parameter estimates**'. This has been shown to be a good standard practice for MCMC-based estimation. Finally, we've selected the '**Logit**' as the link function. Because the logit link is a monotonic transformation (as opposed to the sin link), evaluation of the moments of the posteriors is somewhat easier (and less prone to error). Finally, we've added the word '`MCMC`' to the model name, to indicate that the model will be estimated using MCMC (although this isn't really necessary, since the model results are not added to the browser).



Once you've made these changes, click the '**`OK to Run`**' button. This will spawn a new window (shown below) which will allow you to specify the '**`Markov Chain Monte Carlo Parameters`**'.



We'll quickly go down the list of those parameters we're going to need to consider when using MCMC for the binomial survival example. First, the '**`Random Number Seed`**'. It defaults to 0, meaning, the random numbers used to generated the samples will be different for each run. For almost all purposes, this is entirely acceptable, so we leave the seed at the default value of 0 in most cases. The purpose of this entry is to allow you to specify a random number seed if you want to duplicate results from a previous analysis.

Next, 3 boxes indicating the number of 'samples' we want to make. The specific details concerning these options are detailed in the Addendum to this appendix, but briefly:

Number of 'burn in' samples – the Metropolis-Hastings algorithm used by **MARK** takes random samples from the posterior distribution. Typically, initial samples from the Markov chain are not completely valid because the chain has not stabilized. The 'burn in' samples allow you to discard these initial samples.

Number of 'tuning' samples – MCMC is based on acceptance/rejection of a 'proposed' value, drawn from a 'proposal distribution' – typically $\mathcal{N}(0, \sigma)$, where σ is chosen estimated to give a 40-45% acceptance rate. That is, σ is estimated during the 'tuning' phase to accept the new proposal 40-45% of the time.

Number of samples to store – After the initial 'burn in' period (see above), samples from the posterior distribution are saved to enable computation of summary statistics describing this (posterior) distribution. [*Note*: thinning the sample is not an option within **MARK**, but can be done after the fact using **SAS** or **R**.]

For smaller problems, these defaults are generally sufficient. For larger data sets, and more complex models, you will typically need to increase the size of the various samples (in particular, the '**Number of samples to store**').

Next, a box to let you specify the name of binary file to store samples (it defaults to `MCMC.BIN`). The samples are saved in a binary file (meaning, you can't simply open up the file in an ASCII editor). The binary file can be read, however, with a **SAS** code or an **R** code to perform more sophisticated analysis than are available from **MARK**. Example **SAS** and **R** code is provided in the **MARK** helpfile. This box is followed by another which lets you specify the name of a .CSV file to storE summary data (means, medians, percentiles...) from the sample data (it defaults to `MCMC.CSV`). This file can be opened in Excel, or equivalent spreadsheet software.

Finally, a box to set the default SD of the normal proposal distribution. The default is 0.5. As noted above, for beta parameters that are not a part of hyperdistributions, the default step size to generate the next value of the parameter is generated as a random normal variable with a SD specified in this edit box. Typically, you want to accept the new parameter value about 45% of the time, so you can adjust this SD to approximately obtain this acceptance rate. (Discussed more fully in the Addendum).

The next 3 sections are particularly important. First, the '**Hyperdistribution Specification**'. One of the primary purposes of the MCMC algorithm in **MARK** is to estimate mean and variance of sets of parameters, i.e., estimate the values of μ and σ given a hyperdistribution of a set of beta parameters. Using this edit box, you can specify the number of hyperdistributions that you want to model. If you leave the number of hyperdistributions set to the default of 0, **MARK** will use MCMC to derive estimates of parameters and credibility intervals for those estimates, but will not estimate μ or process variance σ^2. For our re-analysis of the binomial survival data, we want to specify 1 hyperdistribution (i.e., to estimate μ and σ^2 among the set of 10 survival estimates).

As you will observe, once you set the number of hyperdistributions to a value > 0, then two options will be made available ('active'). First, you can choose to model the hyperdistribution means using a design matrix. Checking this check box allows you to model the means of the hyperdistributions with linear models specified in the MCMC hyperdistribution design matrix. For this example, we will not specify a linear model with a design matrix, so we'll leave the box unchecked.

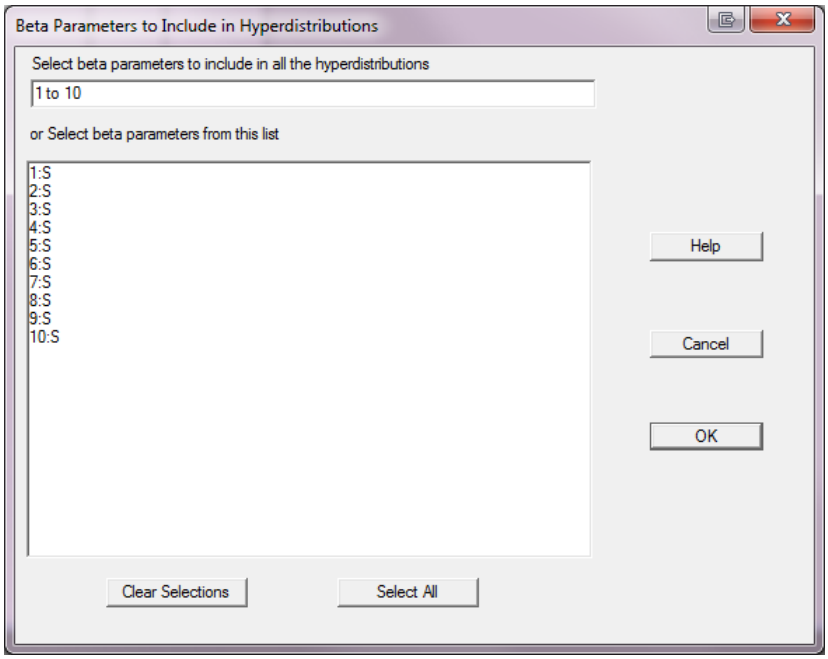
Second, you can specify the structure of the variance-covariance matrix among parameters included in the hyperdistributions. This specification is especially helpful if you are interested in looking at covariance among different parameters in a particular model (e.g., the covariance of S and f in a

Brownie dead recovery model; example scenario 2, on p. E-3). Since that is not our interest here (clearly, since there is only a single parameter type, S_i), we will leave this option unchecked as well.

Next, the specification of '**Prior Distributions for Parameters not included in Hyperdistributions**'. Although the most likely use of the MCMC estimation procedure is to estimate the mean and variance of hyperdistributions, most models in **MARK** include other nuisance parameters in the models. These parameters also require a prior distribution. Three options are provided. The first is to ignore the prior distribution, and never use it to decide whether a new value in the Markov Chain is accepted or rejected. The second option is to specify a default prior distribution, consisting of a normal distribution with the mean and variance provided. All parameters not included in a hyperdistribution will use this normal prior. The third option is to specify the prior distribution for each parameter individually. However, only normal priors are allowed, so you can only specify a mean and standard deviation appropriate for each of the non-hyperdistribution parameters.

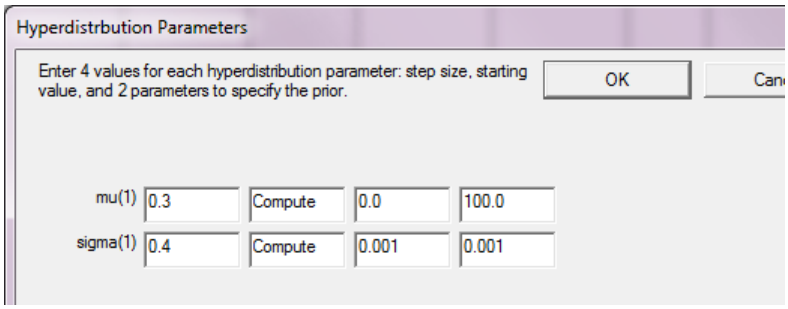
Finally, a section letting you specify various '**Convergence Diagnostics**'. This group of controls determines what diagnostic values will be generated. Checking the '**GIBBSIT sample size**' box will produce a table of estimated sample sizes for the burn-in period and the samples to store. Selecting the '**multiple Markov chains**' option will produce a diagnostic statistic ($\hat{R}$) useful for determining if the Markov chains have adequately sampled the posterior distribution (i.e., if the multiple chains have converged, indicated by $\hat{R} \rightarrow 1.0$). The default is a single chain, which is usually sufficient for simple problems (such as the binomial survival analysis), given enough samples. The **MARK** helpfile should be consulted for further details on these options.

Once you have specified the appropriate values, or want to accept the defaults, click the '**OK**' button to continue. Since we have have specified > 0 hyperdistributions to be modeled (1, for this example), additional dialog windows will request information on these hyperdistributions.



For this example, we have 10 survival parameters ($S_1, S_2, \ldots, S_{10}$), so we enter '`1 to 10`' as the parameters to include in the hyperdistribution. Here, we include all 10 survival parameters. In situations where one or more parameters are confounded, we generally exclude them from the hyperdis-

tributions (similar to excluding confounded parameters in moment-based RE models, as discussed in Appendix D). Click the '**OK**' button. Based on the lists of means and standard deviations selected, a new dialog window appears that is requesting information about how to estimate these hyperparameters.



Four inputs are requested for each hyperparameter. The first edit box is to specify the step size to be used to generate new parameter values with which to sample the posterior distribution. As with the default step size, the goal is to tune the estimation so that approximately 45% of the steps are accepted.

The second edit box is to specify an initial value to start the Markov Chain. The default is '`compute`', which tells **MARK** to compute an estimate from the initial values of the beta parameters. It is generally recommended practice that you should run the model that you want to use for MCMC estimation as a typical **MARK** analysis, so that you can provide initial estimates to start the MCMC estimation.

The third and fourth edit boxes specify the parameters for the *prior* distribution to be used with this hyperdistribution. For mean parameters, a normally distributed prior is used. The third edit box specifies the mean, and the fourth edit box specifies the standard deviation. The default values for μ are mean = 0 and standard deviation = 100, giving a very flat and uninformative prior. For σ parameters, a γ prior is used to model the σ in a transformation: $1/\sigma^2$ is assumed to be distributed as a γ distribution with parameters α (third edit box) and β (fourth edit box). Again, the defaults of $\alpha = \beta = 0.001$ result in a very flat, uninformative prior. In other words, the defaults specify uninformative 'flat' priors for the two hyperparameters, μ and σ.

Once you click '**OK**', you'll be asked to specify '**`starting values`**', which you can retrieve from the general model $\{S_t\}$, which is already in the browser (remember – retrieving from a model in the browser only works if the model you're retrieving was run with the same link function as we're using here – logit). Once you have set the starting values, and clicked '**OK**', **MARK** will spawn a command windows ('DOS box'), which will show you the progress of the MCMC sampling (i.e., which chain **MARK** is working on, which iteration). In practice, you will quickly discover that this stage can (and frequently is) be the 'rate-limiting step' for using MCMC in **MARK** (or any other software), especially for complex problems (which in the context of **MARK**, can also mean 'lots of parameters, and lots of data'), where even the default 10,000 samples (plus 5,000 for burn-in and tuning) can take a very long time. There isn't much you can do easily to speed things up – in general, the only guaranteed way to speed things up is to get a faster computer.

For our re-analysis of the binomial data, this is a probably a moot point. **MARK** will probably generate the 15,000 total samples in only a few seconds on even a 'mediocre' computer. Once **MARK** has finished generating the samples, the resulting output is not stored in the results browser, or in the results file. Rather, the output, containing various summary statistics, is placed in an editor window, as well as a .CSV file that can be opened in Excel, or equivalent spreadsheet software. Summaries are provided for each of the beta parameters, hyperdistribution parameters, and real parameters are the mean, standard deviation, median, and mode, plus the percentage of trials when a new value was accepted (labeled

as the proportion of jumps accepted). In addition, the frequency percentiles of 2.5, 5, 10, 20, 80, 90, 95, and 97.5, as well as 80%, 90% and 95% highest posterior density intervals, are listed. These values can be used to create credibility intervals for the parameters (discussed in more detail, below). In addition, as noted, the binary output file is available to use for additional, more sophisticated analysis, with the **SAS** code or **R** code provided in the **MARK** helpfile.

When you scroll through the output in the editor window, you'll eventually come to the summary statistics for the posterior samples. Here they are for the binomial survival example (*note*: MCMC is based on taking random samples from the posterior – the word 'random' explicitly indicates that the values you get in your data summaries are unlikely to precisely match those reported here. But, they should at least be relatively close).

LOGIT Link Function Parameters of {S(t) -- general model - MCMC}

Parameter	Mean	Standard Dev.	Median	Mode	Jumps(%)	Jump Size
1:S	0.0506165	0.2408051	0.0194106	-0.0304983	60.8	0.15190
2:S	-0.1515116	0.2266404	-0.1290978	-0.1123697	52.9	0.21350
3:S	-0.0654622	0.2018869	-0.0639737	-0.0670698	57.2	0.18009
4:S	-0.0950808	0.2106172	-0.0865177	-0.0526296	61.5	0.15190
5:S	-0.0654535	0.2035320	-0.0639865	-0.0495760	52.7	0.21350
6:S	-0.1832240	0.2385719	-0.1516941	-0.1080016	61.5	0.15190
7:S	0.0247361	0.2254097	0.0029225	-0.0090903	56.9	0.18009
8:S	-0.1247292	0.2114063	-0.1103825	-0.0888076	52.9	0.21350
9:S	-0.0056581	0.2128594	-0.0186380	-0.0242409	61.2	0.15190
10:S	-0.0070807	0.2126946	-0.0201841	-0.0589582	56.8	0.18009
11:mu(1)	-0.0604595	0.1456781	-0.0616548	-0.0695403	64.6	0.04614
12:sigma(1)	0.1792769	0.1469953	0.1350572	0.0411071	56.9	1.11059
13:-2log Likelihood	345.91936	2.7437408	346.10351	346.42924		

-2log Likelihood for means of beta estimates = 342.96482

DIC = 12.763268

At the top of the output (above), it is important to note that what you see here are 'logit link function parameters'. In other words, you're presented with the mean (and SD), median and mode of the distribution of the samples for each parameter (S_i, and the hyperparameters μ and σ) on the logit scale. Meaning, a back-transformation of some sort will be needed to generate parameter estimates on the real probability scale. More on this in a moment. What information do we glean from the output? We see that the mean, median, and mode are different for most parameters, occasionally markedly so. This difference indicates in part that the posterior is potentially not 'nice and symmetrical' (if the distribution was perfectly symmetrical, then the mean, median and mode would be identical). There is clearly some uncertainty on choosing a particular value (mean, median, mode) to generate back-transformed estimates on the real probability scale. In general, using the mean seems to work well, but if in doubt, consult your local Bayesian for their thoughts on the matter.

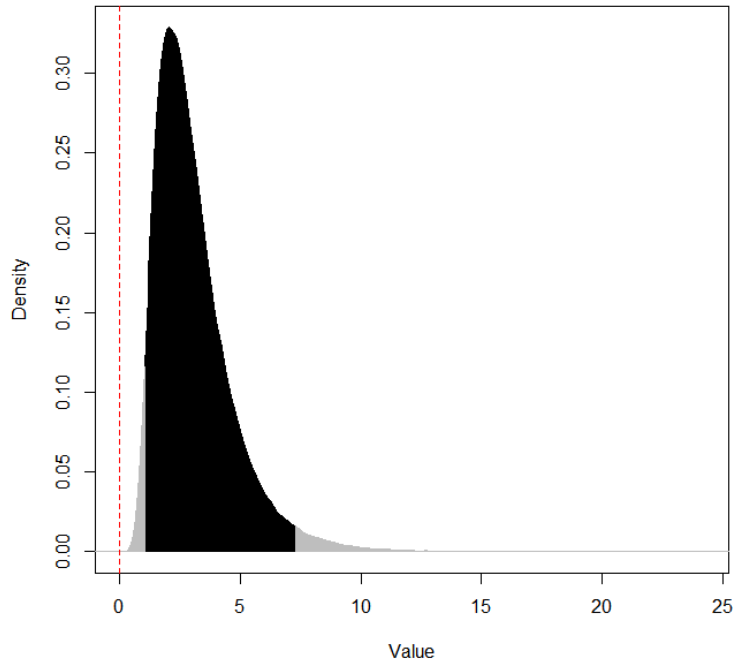
The right-hand column gives the proportion of jumps for each parameter. Ideally, we'd like this percentage to be in the 40-50% range (the proportion of jumps is an index to how efficiently we're sampling the posterior). Some theory has been proposed that 40-50% is 'optimal' for exploring the posterior (optimal being a function of time to convergence, and the ability to explore all regions of the posterior distribution), which **MARK** adopts, in a fashion. As noted on the preceding page, you can 'tweak' the jump size by changing the step size **MARK** uses to generate new parameter values with which to sample the posterior. In addition to the summary statistics for each of the parameters, **MARK** also reports the estimated $-2\ln\mathcal{L}$ for the model, including the $-2\ln\mathcal{L}$ based on the means of the beta estimates (where near equality of the two values indicates that using the mean is probably acceptable for this example).

Finally, **MARK** reports the DIC (Deviance Information Criterion) for the model, which in theory could be used for model selection, model averaging, and other purposes. The DIC is a hierarchical modeling generalization of the AIC (Akaike information criterion) and BIC (Bayesian information

criterion). The DIC is calculated as the sum of the model deviance – the difference in $-2\ln\mathcal{L}$ of the current model and $-2\ln\mathcal{L}$ of the saturated model – and the effective number of parameters. Clearly, the DIC has the same 'fit plus parameter penalty' structure as do the AIC and BIC. And, like the AIC and BIC, it is an asymptotic approximation as the sample size becomes large. It is only valid when the posterior distribution is approximately multivariate normal. Because of current challenges (and differences of opinion) on how best to handle model selection and averaging in a Bayesian context, the DIC is printed in the output for informational purposes only. Use at your own risk.

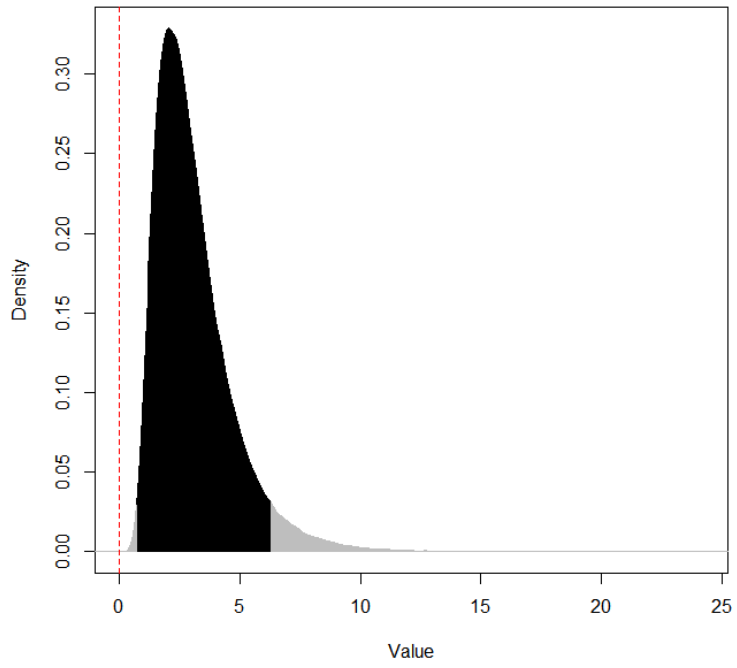
Immediately below the summary statistics of the parameters on the logit scale are the percentile tabulations for each parameter – useful in generating (say) a 95% credibility interval, on the logit scale. Two different types of percentiles are generated: the first based on simple frequency percentiles for the posterior distribution (i.e., 2.5%, 5%,...,95%, 97.5%), and a second based on the highest posterior density (HPD). For unimodal, (more-or-less) symmetric posterior distributions, the frequency percentile- and HPD-based credible intervals won't differ much. However, the HPD is potentially the more useful basis for specifying a credible interval when the posterior distribution is skewed, or not symmetric. In brief, the HPD is an interval that spans a specified percentage of the distribution (say, 95%), such that every point inside the interval has a higher plausibility than any point outside the interval. For a simple, univariate parameter θ with a unimodal posterior probability density, calculation of a simple HPD is relatively straightforward: (i) sort the posterior values for θ in ascending order, (ii) for $j = 1, 2, \ldots N-[(1-\alpha]N]$, when N posterior samples, compute $N(1-\alpha)\%$ credible intervals (e.g., for a posterior consisting of $N = 1{,}000$ samples, for an $\alpha = 0.95$ (i.e., for a 95% HPD), you construct $1{,}000 \times (1 - 0.95) = 50$ intervals), where the credible interval is $[\theta_j, \theta_{(j+[(1-\alpha)N]}]$. Finally, (iii) the HPD interval is the one with the smallest interval width among all credible intervals generated in step (ii).

Let's take a closer look at how these intervals differ. Consider a large sample of log-normal random deviates, with mean and standard deviation on the log scale of 1.0 and 0.5, respectively. The following is a density plot of the sample data – clearly, this distribution is not symmetrical. The left- and right-hand grey tails are the 2.5% percentiles, constructed such that the area of both tails is the same, with the dark (black) area representing 95% of the probability mass:



The 95% frequency percentile shown in the figure [1.0672, 7.2661]. But, consider the probability of $\theta = 1$. Based on the density plot, $\theta = 1$ is far more likely to occur than $\theta = 6$, and yet $\theta = 6$ is included in the frequency percentile-based credibility interval, whereas $\theta = 1$ is not. Clearly, this does not inspire much confidence (pun partially intended...) in our 95% credibility interval for θ based on simple frequency percentiles.

Contrast this with a credibility based on highest posterior density (HPD), indicated in the following density plot. Here, the grey tails represent those values of θ where our belief in the credibility of that values is < 5%, whereas the dark, black area represents the values of θ for which the sum of our belief in those points (i.e., the area under the curve) is 95%.



The 95% HPD interval shown in the preceding plot, [0.7027, 6.2893], is clearly quite different than the one based on frequency quantiles (above). The important distinction here is that the probability at either the left-hand or right-hand 'border' between the grey and black regions is identical. In other words, if you drew a horizontal line parallel to the x-axis, it would intersect the probability function at the same point at either 'border'.

Which credibility interval should we report? Well, to some degree, it might seem reasonable to use the HPD generally, because it is often more defensible than the quantile-based interval when the posterior distribution is asymmetrical or (worse) multi-modal. When the posterior is symmetrical and unimodal, then the HPD and quantile-based intervals are effectively identical in practice.

The biggest potential problem for the HPD is its lack of invariance under transformation. And, neither the frequency percentile-based probability interval or the HPD interval take the prior distribution into account. Yet another, computationally more challenging approach which does take the prior into account is based on the 'lowest posterior loss' (LPL) functions. [At present,calculation of LPL credibility intervals has not been implemented in **MARK**.] See Bernardo (2005) for the rather messy details.*

* Bernardo, J. M. (2005). Intrinsic Credible Regions: An Objective Bayesian Approach to Interval Estimation. *Sociedad de Estadistica e Investigacion Operativa*, **14**, 317-384.

Immediately following the frequency percentile- and HPD-based credible intervals in the **MARK** output are the estimates and moments for the individual structural parameters (but note – not the hyperparameters), back-transformed to real probability scale.

Here are the reconstituted estimates for the binomial survival example:

Real Function Parameters of {S(t) -- general model - MCMC}

Parameter	Mean	Standard Dev.	Median	Mode
1:S	0.5122326	0.0587721	0.5048525	0.4923201
2:S	0.4628398	0.0549111	0.4677703	0.4718443
3:S	0.4838063	0.0497401	0.4840120	0.4831943
4:S	0.4765415	0.0516942	0.4783840	0.4868165
5:S	0.4838067	0.0501470	0.4840088	0.4876349
6:S	0.4551692	0.0574450	0.4621490	0.4729862
7:S	0.5059414	0.0552734	0.5007306	0.4977442
8:S	0.4692801	0.0517626	0.4724324	0.4739214
9:S	0.4984989	0.0523712	0.4953406	0.4938752
10:S	0.4981472	0.0523627	0.4949541	0.4852393

Now, it is critical to remember that for this analysis, we are estimating a model where we've specified a hyperdistribution for the individual S_i, with hyperparameters μ and σ. In other words, this is a mean (intercept only) random effects model. Thus, the reported estimates for S_i (above) are, in fact, shrinkage estimates, $\tilde{S}_i$, where the estimates are 'shrunk' towards the common mean of the hyperdistribution. Shrinkage estimates are introduced and discussed in detail in Appendix D.

Also, note that what is reported for each parameter is the mean of the posterior, back-transformed from the logit to the real probability scale. For example, from the preceding table of real function parameters, we see that the estimated mean for parameter S_1 on the logit scale was $\tilde{S}_{1,\text{logit}} = 0.0506165$.

If we back-transform from this value on the logit scale to the real probability scale, we get

$$\tilde{S}_1 = \frac{e^{0.0506165}}{1 + e^{0.0506165}} = 0.51265$$

which is close to what **MARK** reports for $\tilde{S}_1 = 0.51223$ above. Why the slight difference? The answer is because **MARK** does not simply back-transform the estimate for $\tilde{S}_{1,\text{logit}} = 0.0506165$ (as we have done here) – rather, **MARK** takes the entire posterior sample on the logit scale, and back-transforms it to the real probability scale, and then takes the mean of this back-transformed distribution.

How do these back-transformed estimates of the individual $\tilde{S}_i$ compare to estimates derived using maximum likelihood and the 'method of moments' approach (from p. E-6)? In the following table (E.2), we compare the first 5 estimates from both methods (rounded to 3 significant digits). As we can see, the results from both approaches match pretty well, which they should, given we used a non-informative prior for both hyperparameters (see the following - `sidebar` -).

Table E.2: *Comparison of shrinkage estimates from the ML 'methods of moments' (Appendix D) with shrinkage estimates from MCMC analysis of the binomial survival data.*

	i				
method	1	2	3	4	5
$\tilde{S}_i$ (ML)	0.548	0.431	0.482	0.465	0.482
$\tilde{S}_i$ (MCMC)	0.512	0.463	0.483	0.477	0.484

begin sidebar

ML estimates and MCMC

In the preceding example, we fit a model where we specified a hyperdistribution for the S_i, with hyperparameters μ and σ. The estimates of the S_i values were, therefore, shrinkage estimates, $\tilde{S}_i$. How would we use MCMC to generate what are equivalent to ordinary ML estimates? Simple enough – we simply re-run the MCMC analysis without specifying the hyperdistribution on the individual S_i, with a non-informative prior on the beta parameters.

However, we need to think a bit about how **MARK** handles priors for parameters which are not included in a hyperdistribution – meaning, in this case, all the parameters. These parameters also require a prior distribution. As noted earlier, three options are provided in **MARK**. The first is to ignore the prior distribution, and never use it to decide whether a new value in the Markov Chain is accepted or rejected. The second option (which is selected by default) is to specify a default prior distribution, consisting of a normal distribution with the mean and variance provided. All parameters not included in a hyperdistribution will use this normal prior. The third option is to specify the prior distribution for each parameter individually. However, only normal priors are allowed, so you can only specify a mean and standard deviation appropriate for each of the non-hyperdistribution parameters.

Care must be taken in what prior is used for parameters not included in a hyperdistribution. A mean of zero is appropriate for most parameters. However, a very large standard deviation will result in a back-transformed logit value with a 'U-shaped' distribution, i.e., the real parameter value is much more likely to take on the values close to zero or close to 1. A standard deviation of 1.5 results in a back-transformed distribution that as about 95% of the probability between [0.05, 0.95], so is actually results in a pretty reasonable prior distribution on the real scale. Further, some consideration must be given to how the prior distributions of a function of the beta parameters will interact. For example, the intercept and slope of a trend model might not be appropriately specified as $\mathcal{N}(0, 1.5)$ priors, depending on how the beta parameters are expected to change over the range of the data.

The other consideration about the prior is that if it is non-informative, then the mode of the posterior posterior distribution, as generated by MCMC, should be identical with the MLE. This can be easily demonstrated by algebra. Consider estimation of a simple binomial (say, a coin toss experiment). If the probability of tossing a head is θ, then (from Chapter 1) the probability distribution is given by the binomial distribution, i.e.,

$$f(x \mid \theta) = \frac{n!}{x!(n-x)!}\theta^x(1-\theta)^{n-x}.$$

As discussed briefly in Chapter 1, it is clear that the likelihood given this expression is maximized at $\theta = x/n$.

In a Bayesian context, we generally imagine placing a prior distribution on the parameter θ. Since the parameter θ lies within the interval [0, 1], then an appropriate prior is also bounded on the interval [0, 1]. In the absence of prior information (or belief) we might consider a non-informative ('flat') prior which makes all possible values of θ equally likely. For convenience, the Bayesian often uses a general prior distribution which makes it convenient to modify the prior to reflect a range of prior beliefs. One common approach is to take the beta distribution (which is conjugate for the the binomial) such that $P(\theta) \propto \theta^{\alpha-1}(1-\theta)^{\beta-1}$. The shape variables α and β are the prior parameters, and can be chosen to reflect differing prior beliefs. If $\alpha = \beta = 1$, then the distribution is flat (uniform) over the interval [0, 1] (Fig. E.4). Note also that whenever $\alpha = \beta$, the prior distribution is symmetric around $\theta = 0.5$ (it is only symmetric and flat if $\alpha = \beta = 1$).

If we adopt a beta prior on θ, then the posterior is given as

$$\pi(\theta \mid x) \propto f(x \mid \theta)p(\theta) \propto \theta^{x+\alpha-1}(1-\theta)^{n-x+\beta-1}$$

Thus, we see that the posterior is itself in the form of a beta distribution, with parameters $(x + \alpha)$ and $(n + b - x)$.

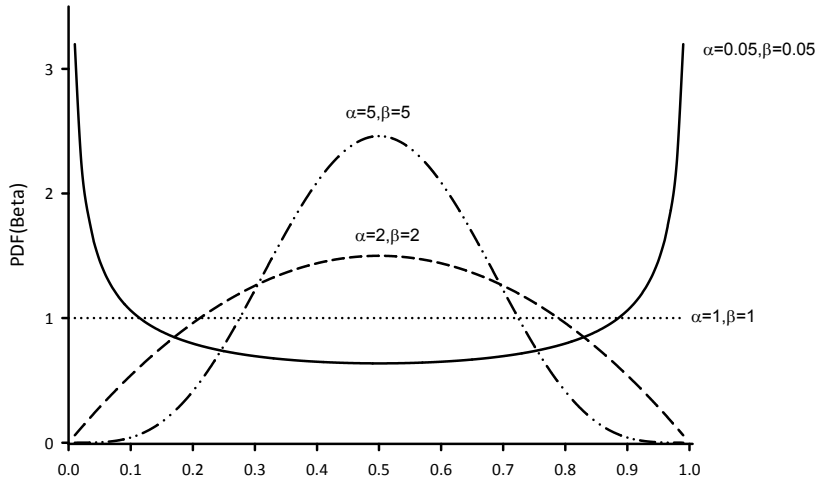


Figure E.4: *Shape of the beta probability distribution for different values of the shape parameters α and β. The distribution is symmetric for $\alpha = \beta$, and is uniform for $\alpha = \beta = 1$.*

By the method of moments (see first few pages of Appendix B), we can show that this posterior distribution has a mean of

$$\frac{x+\alpha}{n+\alpha+\beta},$$

and a mode of

$$\frac{x+\alpha-1}{n+\alpha+\beta-2}.$$

Now, both the expected mean and mode of the posterior appear quite different than the MLE of (x/n). But, look closely at the expressions for the expected mean and mode of the posterior. You should see that, in fact, the MLE is embedded in the two expressions.

Why is this important? It is important because this suggests that the Bayesian estimate for θ will differ from the ML estimate for θ as a function of the values of α and β used in the prior. Given that, what happens if we use a flat, non-informative prior, $\alpha = \beta = 1$? If we do, then we see that the expected mean of the posterior is

$$\frac{x+\alpha}{n+\alpha+\beta} = \frac{x+1}{n+1+1} = \frac{x+1}{n+2},$$

while for the mode,

$$\frac{x+\alpha-1}{n+\alpha+\beta-2} = \frac{x+1-1}{n+1+1-2} = \frac{x}{n}.$$

In other words, for a 'flat', non-informative prior, the mode of the posterior and the MLE are *identical*. Also, note that as the sample size increases (i.e., as x and n both increase), then the posterior mean will converge on the mode and the MLE. (For completeness, we also note that as the sample size gets larger, the variance of the posterior gets smaller – just as with ML estimation).

So, if we use a non-informative prior, the parameter estimates from **MARK** should match the MLE pretty closely. The following table (E.3, top of the next page) compares the MLE and MCMC estimates for the first 5 parameters. As expected, the estimates are close between the two methods.

Table E.3: *Comparison of maximum likelihood (ML) estimates to estimates from MCMC analysis of the binomial survival data.*

	i				
method	1	2	3	4	5
$\tilde{S}_i$ (ML)	0.640	0.360	0.480	0.440	0.480
$\tilde{S}_i$ (MCMC)	0.635	0.367	0.474	0.450	0.477

end sidebar

Finally, at the bottom of the output, you'll see some 'snippets' of **SAS** and **R** code, respectively:

```
SAS Macro Variable Values:
%let ncovs=10; * Number of beta estimates;
%let nmeans=1; * Number of mean estimates;
%let ndesigns=0; * Number of design matrix estimates;
%let nsigmas=1; * Number of sigma estimates;
%let nrhos=0; * Number of rho estimates;
%let nlogit=10; * Number of real estimates;
%let MCMCfile='MCMC.BIN'; * Name and path to the MCMC.BIN file;


R Variable Values:
ncovs <- 10; # Number of beta estimates
nmeans <- 1; # Number of mean estimates
ndesigns <- 0; # Number of design matrix estimates
nsigmas <- 1; # Number of sigma estimates
nrhos <- 0; # Number of rho estimates
nlogit <- 10; # Number of real estimates
filename <- "MCMC.BIN"; # Name and path to the MCMC.BIN file
```

These bits of code contain the values for different variables you need to 'copy and paste' into the **SAS** or **R** code to further process the sample data. This code can be found in the **MARK** helpfile.

One particularly helpful stage of further analysis of the sample data is the visualization of the density and trace of the posterior samples. For example, here (Fig. E.5) is the time-series (trace) for a single chain for β_6 (based on 25,000 samples), after dropping the tuning and 'burn-in' samples:

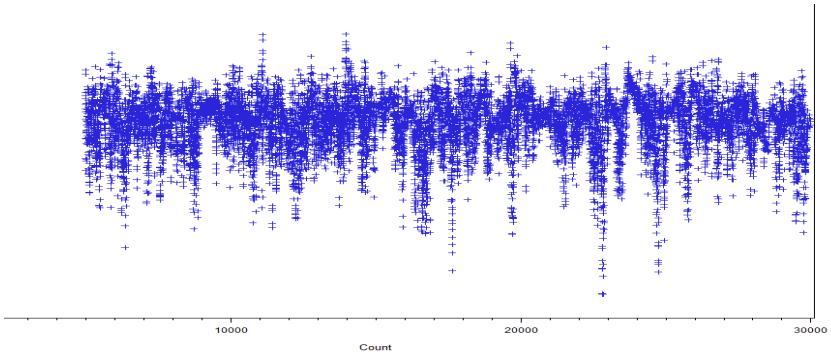
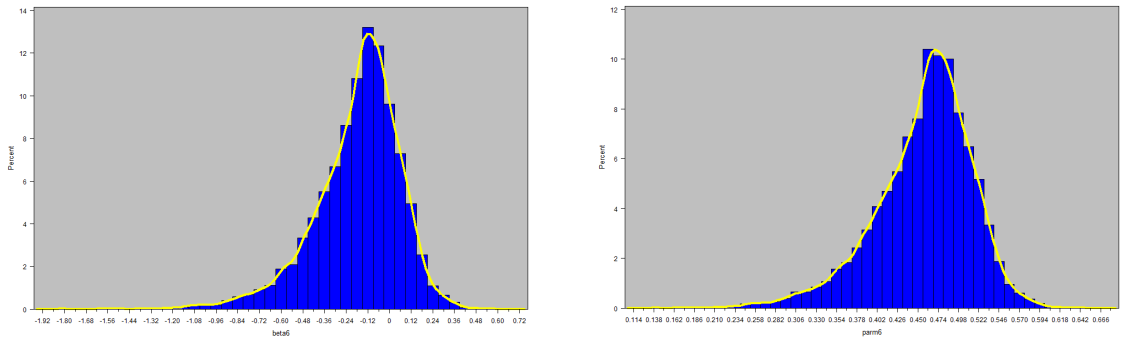


Figure E.5: *Time-series 'trace' of posterior samples for β_6, on the logit scale. Trace based on* 25,000 *samples.*

We expect that the trace should look like what has been described as a 'fuzzy lawn' – roughly an equal number of samples above and below some imaginary horizontal line drawn through the central mass of the samples. More specifically, we hope that the sampler doesn't seem to get 'stuck' at any point (i.e., that it is able to explore the entire domain of the posterior distribution). Based on this somewhat subjective criterion, the trace for β_6 (Fig. E.5) would probably be considered as 'satisfactory'. [As an exercise, try changing the step size parameters, and looking at see what this does to the trace. As noted, we try to set the step size to balance efficiency – moving around the posterior in as few samples as possible – with the ability to actually 'reach' all parts of the posterior. If you try changing the step size, you'll develop a fair intuition of how this all works.]

In the following figure (E.6), we compare the sample density for β_6 (on the logit scale) and the back-transformed parameter $\tilde{S}_6$ (on the real probability scale).



(a) *density plot from posterior sample for β_6 (logit)* **(b)** *density plot from posterior sample for $\tilde{S}_6$ (real)*

Figure E.6: *Density plots for MCMC samples for β_6 (on the logit scale), and the back-transformed value S_6 (on the real probability scale).*

Here, we are primarily interested in establishing that the density plots are unimodal. If the plots have > 1 mode, then there is some chance the sampler was 'stuck' in a local minimum, and has biased our parameter estimates (this is discussed in some detail with respect to multi-state models (Chapter 10), where such multi-modal posterior distributions are regrettably common). As discussed 9in more detail earlier, we can also attributes of the 'width' of the density plot (wide, narrow) to establish estimated parameter precision.

These are only 2 of a couple of graphical representations of the MCMC data. In addition, there are a very large (and seemingly ever-growing) number of diagnostic tools (some graphical, some analytical) to allow you to more fully explore the MCMC sample data. In this Appendix, we will be working primarily with examples where the posterior distribution has 'good properties' (although we do include one example where the results are somewhat 'problematic', reflecting some of the fairly common challenges you might face in practice). Working with MCMC (in **MARK**, or otherwise) inevitably means you'll spend considerable time evaluating and working with these tools.

E.1.2. estimating the hyperparameters μ and σ

OK, so we see how we can use MCMC to derive estimates for the annual survival parameters, either ML estimates, $\hat{S}_i$, or shrinkage estimates, $\tilde{S}_i$. The most obvious reason we might choose to do so is because the MCMC approach generates the percentiles we can use to specify a 95% credibility interval. Not only is this arguably more 'defensible' in the conceptual sense (since the credibility interval is perhaps more

meaningful, and certainly less confusing, than the more familiar 'frequentist' 95% confidence interval, since the latter is based on expectations from a theoretical number of replicates of the data), but the credible interval may actually perform better (or at least as well) than the profile likelihood-based CI's for parameters estimated near the [0, 1] boundaries.

But our primary interest here is the estimation of the hyperparameters for the random effects model. Recall from p. E-6 that, on the real probability scale, $\hat{\mu}_{\mathrm{MOM}} = 0.482527$, and $\hat{\sigma}_{\mathrm{MOM}} = 0.0441609$ (where the 'MOM' subscript indicates that the estimates were derived using the 'method of moments' approach to variance components analysis discussed in Appendix D). From the results of our MCMC analysis (p. E-11), we see that $\hat{\mu}_{\mathrm{MCMC}} = -0.0604595$, and $\hat{\sigma}_{\mathrm{MCMC}} = 0.1792769$, on the logit scale (here, the 'MCMC' subscript indicates that the estimates were derived using the MCMC approach).

What happens if we take these estimates from the MCMC analysis, and back-transform from the logit scale to the real probability scale? We'll start with μ – recall that μ is our best estimate of the overall mean of the parameter:

$$\hat{\mu} = \frac{e^{-0.0604595}}{1 + e^{-0.0604595}} = 0.48489$$

which is close to the value estimated using the 'method of moments' approach ($\hat{\mu}_{\mathrm{MOM}} = 0.482527$). So, it appears straightforward to derive our estimate of the overall mean μ, by taking the estimate from the MCMC analysis on the logit scale, and back-transforming to the real probability scale.

What about the estimate of process variance, σ? What happens if we simply back-transform $\hat{\sigma}_{\mathrm{MCMC}} = 0.1792769$ from the logit to the real probability scale?

$$\hat{\sigma} = \frac{e^{0.1792769}}{1 + e^{0.1792769}} = 0.5447$$

which is clearly not close (not even remotely) to the estimate from the 'method of moments' approach ($\hat{\sigma}_{\mathrm{MOM}} = 0.044161$), which we accept to be 'correct' for these data (see Appendix D).

There are several issues to consider here. First, perhaps we shouldn't be trying to back-transform at all, and should evaluate our estimates directly on the logit scale on which they are estimated. Often the logit scale is the biologically relevant scale at which to work. The logit scale is more likely to provide a linear scale to model the effects of environmental covariates, e.g., precipitation or temperature. However, this sidesteps the objective of comparing the estimates of σ from MCMC with those generated by the 'method of moments' random effects approach. While we might be satisfied and generally safe using estimates of process variance from the moments-based approach, that particular method doesn't apply well to complex models, and we would like to be able to use the more flexible MCMC approach in such cases. Moreover, one of the uses of process variance is in analysis stochastic projection models (as described in the introduction to Appendix D), which are projected using transition probabilities on the real [0, 1] probability scale, not the logit scale.

A second possibility might be that we're not properly accounting for the effects of the transformation on the estimated variance on the transformed scale. If you've worked through Appendix B, it might seem reasonable to consider using the Delta method to estimate the variance μ after transformation from the logit to real probability scale.

From Appendix B, we write the transformation function $f(\mu)$ as

$$f(\mu) = \frac{e^{\mu}}{1 + e^{\mu}}$$

Thus, to first order,

$$\begin{aligned}\widehat{\text{var}} &\approx (f'(\mu))^2 \sigma_\mu^2 \\ &= \left(\frac{\partial f(\mu)}{\partial \mu}\right)^2 \widehat{\text{var}}(\hat{\mu}^2) \\ &= \left(\frac{e^{\hat{\mu}}}{1+e^{\hat{\mu}}} - \frac{(e^{\hat{\mu}})^2}{\left(1+e^{\hat{\mu}}\right)^2}\right)^2 \widehat{\text{var}}(\hat{\mu}^2)\end{aligned}$$

From p. 11, we see that $\hat{\mu}_{\text{MCMC}} = -0.06046$. Now, what should we use for the estimate of the variance of μ? There are 2 values in the output on p. E-11 that you might consider for the variance term in the Delta approximation. First, we see that the variance for the posterior for the parameter μ is estimated as $0.145678^2 = 0.0212218$. We also have the estimate of σ as a parameter itself, $\hat{\sigma} = 0.17928$, such that $\widehat{\text{var}}(\hat{\sigma}) = 0.17928^2 = 0.03214$. Let's try the variance of μ (0.0212218) first.

$$\begin{aligned}\widehat{\text{var}} &\approx \left(\frac{e^{-0.06046}}{1+e^{-0.06046}} - \frac{(e^{-0.06046})^2}{\left(1+e^{-0.06046}\right)^2}\right)^2 (0.0212218) \\ &= 0.0013240\end{aligned}$$

Thus, $\widehat{\text{var}}(\hat{\mu})$ on the logit scale would be approximated as $\sqrt{0.0013240} = 0.03214$. While this is considerably closer to the estimate from the 'method of moments' approach (0.04416) than our naive back-transformed estimate of 0.5447, there is still a fair discrepancy (almost 30% difference) between the estimates.*

Now, let's repeat the calculation, but this time, using the estimate of $\widehat{\text{var}}(\hat{\sigma}) = 0.17928^2 = 0.03214$ for the variance term.

$$\begin{aligned}\widehat{\text{var}} &\approx \left(\frac{e^{-0.06046}}{1+e^{-0.06046}} - \frac{(e^{-0.06046})^2}{\left(1+e^{-0.06046}\right)^2}\right)^2 (0.03214) \\ &= 0.0020051\end{aligned}$$

and, thus, our estimate of $\hat{\sigma}$ on the real probability scale would be approximated as $\sqrt{0.0020051} = 0.0448$. We are rather pleased to observe that this value is close to the estimate from the 'method of moments' analysis (0.0442), differing only in the fourth decimal place.

However, before we become overly satisfied, its important to understand 'why' this seems to have worked. The key is in remembering that μ and σ are being estimated as parameters, and while there is uncertainty in the estimate of each, such that each has its own sampling variance (based on the distribution of MCMC samples for each parameter), the estimate of $\hat{\sigma}$ (as the mean of the posterior for the σ parameter) is in fact the best estimate of the random variation of the annual S_i around the mean, μ, which is itself estimated as the mean of the posterior for the μ parameter. So, the correct value to use for the variance in the Delta approximation is the square of the estimate of σ itself.

* In fact, this estimate of 0.03214 is approximately equal to the estimate for σ you would obtain if (i) you took the entire posterior sample, back-transformed it from the logit scale to the real probability scale, and then (ii) estimated σ from this back-transformed distribution. If you think hard about what is meant by the estimate of σ you used in the Delta transformation to generate the value of 0.03214, you should be able to see the reason for this.

Does this approximation always work? As noted in Appendix B, the Delta method assumes that the transformation function is effectively linear in the region where most of the data reside. For some parameters, especially those near the 0 or 1 boundaries, this may not necessarily be the case. Further, in the present example, we used a non-informative 'flat prior'. If instead we had used an informative prior – in particular, a prior that was asymmetrical over the [0, 1] interval – then it is quite likely that the Delta method may not work particularly well.

begin sidebar

estimating σ by simulation

As noted in Appendix B, where we introduce the Delta method approximation to estimating the variance of functions of one or more parameters (as we just applied, above), another approach to estimating the variance is to use numerical simulation or bootstrapping. Such numerical methods are less convenient in many instances than the Delta method, but are generally less susceptible to violation of some of the necessary assumption required for the Delta method to perform well.

Here, we introduce a simple approach here for estimating the process variance σ^2, based on MCMC sample data. We will demonstrate the method using the binomial survival example. Recall that the 'method of moments' approach generated estimates of $\hat{\mu} = 0.48253$ and $\hat{\sigma} = 0.04416$. Recall also that our MCMC estimates of μ and σ on the logit scale (based on the mean of the posterior distribution for both parameters) were $\hat{\mu}_{\text{MCMC}} = -0.06046$ and $\hat{\sigma}_{\text{MCMC}} = 0.17928$.

What we're going to do is simulate data on the logit scale, given estimates of μ and σ, back-transform the simulated data from the logit scale to the real probability scale, and then estimate σ from these back-transformed simulated data. The only challenge is deciding how to simulate data on the logit scale. As noted earlier, **MARK** samples from the posterior based on the assumption of a logit normal proposal distribution. Thus, our approach will be to

- simulate a logit normal sample based on $(\hat{\mu}_{\text{MCMC}} + \mathcal{N}(0, \hat{\sigma}_{\text{MCMC}}))$
- back-transform the simulated logit normal data to the real probability scale
- estimate μ and σ from the back-transformed simulated data

Here is a snippet of **R** code that implements this sequence of steps, using parameter estimates from the binomial survival analysis To facilitate comparison, we also include the estimates of μ and σ from the 'methods of moments' approach (above):

```
# enter parameters from MCMC
  mu=-0.0604595;  sigma_mean=0.1792769;

# for comparison, enter parameters from moments RE estimation
  s=0.482527; sd=0.0441609;

# initialize vector to hold random samples from logit normal
  lnsamp <- vector()

# draw 25000 random samples from logit normal distribution
  for (i in 1:25000) { lnsamp[i] <- mu+rnorm(1,mean=0,sd=sigma_mean); }

# back-transform simulated data from logit -> real
  backtrans <- vector()
  backtrans <- exp(lnsamp)/(1+exp(lnsamp));

# now estimate and assess mean and sigma estimate from back-transformed data

cat("\n Estimation of mu and sigma on back-transformed data from simulated logit normal\n\n")
```

```
cat("This is estimated S (mu): ",mean(backtrans),", compared to moments estimate of ",s,"\n");
cat("This is estimated sigma: ",sqrt(var(backtrans)),", compared to moments estimate
      of ",sd);
cat("\n\n  [Interest here is in how well the estimates from both approaches match...]\n")
```

when we execute this script, we get the following results:

```
Estimation of mu and sigma on back-transformed data from simulated logit normal

This is estimated S (mu):  0.4846203, compared to moments estimate of  0.482527
This is estimated sigma:  0.04444851, compared to  moments estimate of  0.0441609
```

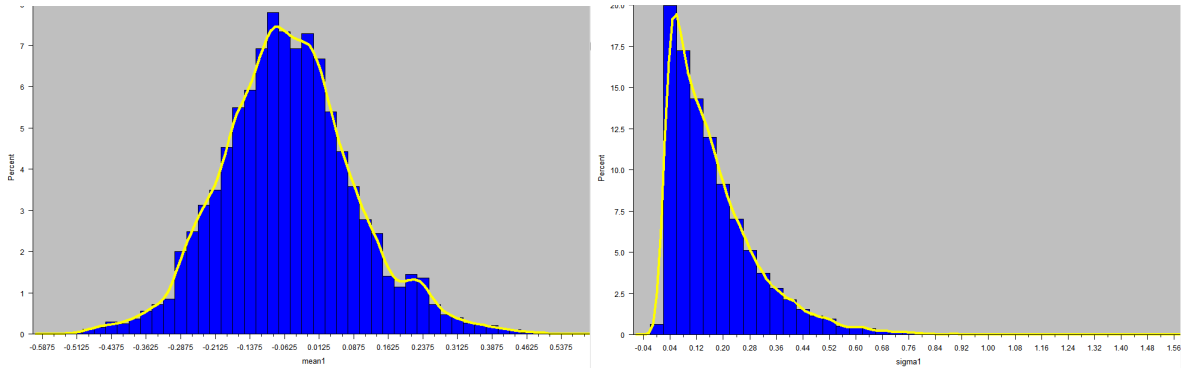
These values estimated from the back-transformed data simulated using estimates of μ and σ from the MCMC analysis are clearly close to the estimates from the 'method of moments' random effects model.

end sidebar

What about credibility intervals for our estimates for $\hat{\mu}$ and $\hat{\sigma}$? Returning to the **MARK** output for the analysis of the binomial survival data, we see that the 95% frequency percentile-based interval reported on the logit scale for $\hat{\mu}$ and $\hat{\sigma}$ were $[-0.3477, 0.2828]$ and $[0.0280, 0.5470]$, respectively. The 95% HPD for $\hat{\mu}$ and $\hat{\sigma}$ were $[-0.3601, 0.2580]$ and $[0.0194, 0.4537]$, respectively.

Which credibility interval should we report? As discussed earlier, we might generally use the HPD, because it is often more defensible than the quantile-based interval when the posterior distribution is asymmetrical or (worse) multi-modal. For the binomial survival data, there is evidence suggesting that the posterior distribution for the mean μ is likely symmetrical, since the reported mean, median, and mode are all quite close: -0.0605, -0.0617, and -0.0695, respectively. In contrast, the mean, median and mode reported for the the variance σ are quite different (0.1793, 0.1251, and 0.0411, respectively), suggesting the posterior distribution for σ is likely asymmetrical.

Our intuition is supported by visual examination of the probability density plots for both parameters. We see that the plot for the mean μ (Fig. E.7a) is indeed quite symmetrical, while for the variance σ, the posterior probability distribution (Fig. E.7b) is quite asymmetrical (strong right-skew) as expected – reporting the HPD might be more appropriate for this parameter.

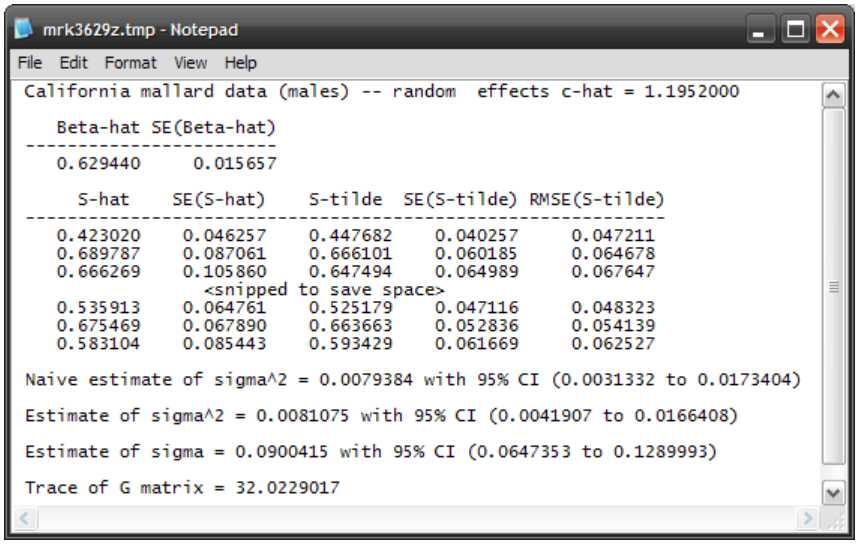


(a) *density plot from posterior sample for μ (logit)* **(b)** *density plot from posterior sample for σ (logit)*

Figure E.7: *Density plots for MCMC samples for μ and σ (on the logit scale).*

E.1.3. Example 2 – California mallard survival re-visited

Here, we will re-visit an analysis of a long-term dead recovery data set based on late summer banding of adult male mallards (*Anas platyrhynchos*), banded in California every year from 1955 to 1996 ($k = 41$) (Franklin *et al.*, 2002). This example was introduced in Appendix D (section D.4.2), where we focussed on the fitting of various random effects models to the mallard recovery data, including a model where survival, S, varied around some unknown mean μ, with unknown process variance σ^2. We referred to this model as $\{S_{\mu,\sigma^2} f_t\}$. If you still have the analysis files for these data that you generated in Appendix D (`california-male-mallard.dbf` and `california-male-mallard.fpt`), this model might still be in your results browser. If not, you can either go back to section D.4.2 in Appendix D and re-create the model, or simply follow our summary of the results for fitting this model to the data (below):



Recall that in Appendix D, we used time-structure for the recovery parameter f. We do so because any constraint applied to f will impart (or 'transfer') more of the variation in the data to the survival parameter S, such that the estimated process variance $\hat{\sigma}^2$ will be 'inflated', relative to the true process variance. In general, you want to estimate variance components using a fully time-dependent model, for all parameters, even if such a model is not the most parsimonious given the data. This is also true if using the MCMC approach we're introducing here.

We see from the results (above) of fitting model $\{S_{\mu,\sigma^2} f_t\}$ to the data, that the estimated mean survival on the real probability scale was $\hat{\mu} = 0.630244$, and process variance was $\hat{\sigma}^2 = 0.0895355^2 = 0.0080166$. Now, let's see if we can replicate this analysis, using the MCMC capabilities in **MARK**. First, retrieve model $\{S_t f_t\}$ in the browser to make it the 'active' model. We are going to re-run this model, using MCMC. Make sure that you check the logit link function. If you built the model using the logit link in the first instance, you can and should check the box telling **MARK** you will provide initial values from that model. If not, take a moment and re-run the model first, using standard ML estimation and the logit link. This is a large (41 years of data), complex dataset, and experience indicates you should probably use the simulated annealing optimization routine (so check the 'alternate optimization' box). Be advised that convergence can take some time.

Once finished, we're ready to start our MCMC analysis, Again, re-run the model, checking the '**`logit`**' link function, '**`Provide initial parameter estimates`**', and (of course) the '**`MCMC Estimation`**' check-boxes.

Next, you'll specify the '**Monte Carlo simulation parameters**'. We'll use the standard defaults for the number of '**tuning samples**' (4,000) and '**burn-in**' (1,000). However, because of the size and complexity of the data set, and the number of parameters being carried in the model, we'll increase the number of MCMC '**samples to store**' from the default of 10,000 to 25,000. Finally, we'll set the '**Number of hyperdistributions**' to 1 (specified by the parameters μ and σ). We'll use default settings for everything else. Once you're ready, click the '**OK**' button.

Next, you'll be presented with a popup window asking you to select which parameters you want to include in the single hyperdistribution. We're interested in estimating μ and σ for the survival parameter, S. Since all S_i and f_i parameters are identifiable in a Brownie model (Chapter 8), we enter '`1 to 41`' (corresponding to survival parameters $\{S_1, S_2, \ldots, S_{41}\}$.

Next, we're asked to enter 4 values specifying the step size, starting value, and the two parameters governing the '**Hyperdistribution parameters**'. We'll accept the defaults (i.e., we'll use default starting values, step size, and the default noninformative 'flat' priors for both μ and σ).

Finally, we're asked to provide '**Initial parameter estimates**' – we'll retrieve starting estimates from model $\{S_t f_t\}$ fit with the logit link (by selecting it from the list of candidate models already in the browser). One the starting values have been retrieved, click the '**OK**' button, and the MCMC sampler will start. Remember – we're doing 30,000 total iterations of the sampler, which for a large and complex data set, will take some time (15-20 minutes on a typical desktop computer).

Once the MCMC samples are finished, **MARK** will dump the summary statistics to the editor. Since our main focus here is on estimation of μ and σ, we'll focus on that part of the output:

```
78:f                    -2.6935079      0.0764739       -2.6928311
79:f                    -2.8313228      0.0794477       -2.8312507
80:f                    -2.9239283      0.0906455       -2.9222336
81:f                    -2.7381426      0.0708654       -2.7372413
82:f                    -2.5601021      0.0746314       -2.5597806
83:f                    -2.4440585      0.0970018       -2.4444575
84:mu(1)                0.5678008       0.0641315       0.5667889
85:sigma(1)             0.3883343       0.0694615       0.3821488
86:-2log Likelihood     65283.015       12.890003       65282.456
   -2log Likelihood for means of beta estimates =   65215.190
   DIC =   595.14177
```

We see that the estimate (on the logit scale) of $\hat{\mu}_{\text{MCMC}} = 0.5678008$, and $\hat{\sigma}_{\text{MCMC}} = 0.3883343$. The back-transform of μ from the logit to the real probability scale yields

$$\hat{\mu} = \frac{e^{0.5678008}}{1 + e^{0.5678008}} = 0.638256$$

which is close to the the estimate $\hat{\mu} = 0.630244$ from the 'methods of moments' approach (above), as expected.

What about $\hat{\sigma}$? Recall from the binomial survival analysis we covered earlier that we can't simply take a back-transform of $\hat{\sigma} = 0.3883343$ from the logit scale to the real probability scale. Instead, we can try either the Delta method, or a numerical simulation approach covered in the preceding - `sidebar` -. For purposes of comparison, we'll use both here. Recall that to first order,

$$\widehat{\text{var}} \approx \left(\frac{e^{\hat{\mu}}}{1 + e^{\hat{\mu}}} - \frac{(e^{\hat{\mu}})^2}{\left(1 + e^{\hat{\mu}}\right)^2} \right)^2 \widehat{\text{var}}(\hat{\mu}^2)$$

From above, we see that $\hat{\mu}_{\text{MCMC}} = 0.5678008$. From the analysis of the binomial survival data, we

know that we use the estimate of σ as a parameter, $\hat{\sigma}_{\text{MCMC}} = 0.3883343$, such that $\widehat{\text{var}}(\hat{\sigma}) = 0.3883343^2 = 0.150805$. Thus,

$$\widehat{\text{var}} \approx \left(\frac{e^{0.5678008}}{1 + e^{0.5678008}} - \frac{(e^{0.5678008})^2}{\left(1 + e^{0.5678008}\right)^2} \right)^2 (0.150805)$$
$$= 0.0080391$$

and, thus, our estimate of $\hat{\sigma}$ on the real probability scale would be approximated as $\sqrt{0.0080391} = 0.089661$, which is close to the estimate from the 'methods of moments' analysis (0.089536).

Using the numerical simulation approach, the estimates of μ and σ are

```
Estimation of mu and sigma on back-transformed scale transformed logit

This is estimated S (mu):  0.633954, compared to moments estimate of  0.630244
This is estimated sigma:  0.0876645, compared to  moments estimate of  0.089536
```

Again, close to the values estimated using the 'methods of moments' approach.

However, in our preceding two examples, mean survival was near the middle of the interval (0.4-0.6). Why is this important? As noted by White *et al*. (2009), the back-transformation of an estimate of σ to the value of σ on the real scale depends on the mean of the distribution. So, an estimate of $\sigma = 0.1$ with a mean of 0 on the logit scale results in a back-transformed real variable with a mean of 0.5 and $\sigma = 0.0025$ with n large. But, an estimate of $\sigma = 0.1$ but with a mean of 4 on the logit scale back-transforms to a real variable with a mean 0.982 and $\sigma = 0.0018$. White *et al*. noted that because of this relationship, '...interpretation of the estimates of the process variance on the logit scale must consider the mean as well'. Note that on the logit scale, $|\mu| \geq 3$ when back-transformed is approaching either the 0 or 1 boundary on the real probability interval over $[0, 1]$. The logit scale on the interval $[\mu = -3, \mu = 3]$ is linear, or nearly so.

So, as a final test before moving on, we'll consider a simulated live encounter mark-recapture (CJS) data set, where true apparent survival alternates from 0.8 to 0.9 in successive years (i.e., $\varphi_1 = 0.8$, $\varphi_2 = 0.9$, $\varphi_3 = 0.8 \ldots$), with encounter probability constant at $p = 0.5$. Thus, the true mean survival μ=0.85, and the true process variance $\sigma^2 \rightarrow 0.0025$. The data set consists of 17 sampling occasions, so 16 intervals for survival, and 16 occasions for recapture. Given that apparent survival alternates between 0.8 and 0.9 in successive years, this means 8 intervals over which survival is 0.8, and 8 intervals over which survival is 0.9. True $\mu = 0.85$, and true $\sigma^2 = 0.00267$. For the simulation, we released 500 newly marked individuals per occasion. To simplify the modeling, and to let us use all 16 estimates of φ, we used model $\{\varphi_t p_.\}$ as our starting, fixed effects model. Normally, we estimate process variance using a fully time-dependent model (see Appendix D for a discussion of this point), but fixing p constant eliminates confounded parameters, and doing so is unlikely to bias results of one method of estimating σ versus the other.

Using these simulated data (contained in `sigmasim.inp`), we estimated the process variance, using both the 'methods of moments' and MCMC approaches. Using the 'methods of moments' approach, $\hat{\mu}_{\text{MOM}} = 0.8517$, which is pretty close to the true mean of 0.85. Estimated $\hat{\sigma}_{\text{MOM}} = 0.0586$ (such that $\hat{\sigma}^2 = 0.00344$), which is not too far off the true process variance of $\sigma^2 = 0.00267$. The discrepancy undoubtedly reflects, at least in part, constraining the encounter probability p to be constant over time. Of more importance here, though, is the MCMC estimate, and if our back-transformation of these estimates from the logit scale to the real scale gives us values close to those from the 'method of moments' approach.

The MCMC analysis (using default values for number of samples), generated estimates of $\hat{\mu}_{\text{MCMC}} = 1.8227$ and $\hat{\sigma}_{\text{MCMC}} = 0.5188$. Using the Delta method, the back-transformed estimate for the mean on the real scale is $\hat{\mu} = 0.8609$, which perhaps not surprisingly is quite close to the true mean of 0.85. For σ, the back-transformed estimate of $\hat{\sigma} = 0.0621$ (such that $\hat{\sigma}^2 = 0.00386$). So, even when true mean survival is relatively close to the boundary, estimates of σ from either the 'method of moments' approach (0.0586) or the MCMC approach (0.0621) are quite close. Whether this holds as μ gets much closer to the boundary needs further study. In such cases, a different link function might be needed (as was the case for some problems considered in Appendix D). In fact, for this example, using the identity link for the MCMC analysis, the reported values for μ (0.8516) and σ (0.0621) are identical to those generated using the back-transformation from parameters estimated on the logit scale.

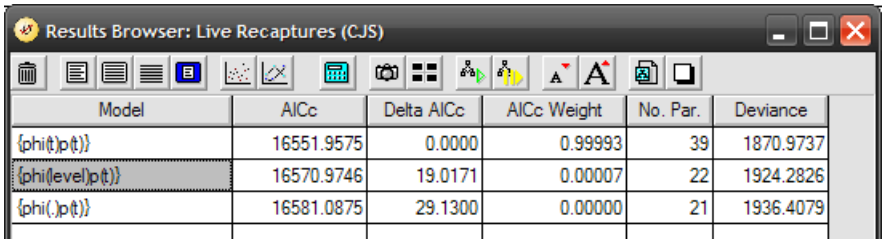
For now, we will defer further discussion of this issue, and proceed under the assumption that, in practice, back-transformed estimates of σ from the logit scale to the real probability scale are robust and unbiased.

E.1.4. Example 3 - environmental covariates re-visited

In the preceding two examples, we explored the mechanics of using MCMC in **MARK** to derive an estimate of process variance, σ^2. If that is all we were interested in, we might choose not to bother with MCMC, when the 'method of moments' approach has been shown to be extremely robust, and is much faster computationally than MCMC.

However, as mentioned in the preamble to this Appendix, there are situations where the 'method of moments' approach is insufficiently flexible to allow us to estimate process variance for certain types of models. We demonstrate this by re-considering an example presented in Appendix D, where apparent survival is believed to vary as a function of a binary environmental covariate (water level). Again, we imagine we have some live encounter (CJS) data collected on a fish population studied in a river that is subject to differences in water level. We hypothesize that annual fish survival is influenced by variation in water level. For this study, we simulated $k = 21$ occasions of mark-recapture data (contained in `level-covar.inp`). Over each of the 20 intervals between occasions, water flow was characterized as either 'average' (`A`) or 'low' (`L`) (more specific covariate information was not available). Here is the time series of flow covariates: {`AAAALLLAAALALALLLLAL`}.

We began our analysis of these data in Appendix D considering 3 fixed effect models for apparent survival, φ: $\{\varphi_t p_t\}$, $\{\varphi_. p_t\}$ and $\{\varphi_{level} p_t\}$. The results from fitting these 3 models to the data are shown at the top of the next page. There was strong evidence for variation over time in apparent survival, but no support for an effect of water level. If you look at the estimates from model $\{\varphi_{level}\}$ for average ($\hat{\varphi}_{avg} = 0.709$, SE = 0.0106) and low ($\hat{\varphi}_{low} = 0.650$, SE = 0.0100), the lack of any support for this model may not be surprising. At least, based on considering water level as a fixed effect.



Model	AICc	Delta AICc	AICc Weight	No. Par.	Deviance
{phi(t)p(t)}	16551.9575	0.0000	0.99993	39	1870.9737
{phi(level)p(t)}	16570.9746	19.0171	0.00007	22	1924.2826
{phi(.)p(t)}	16581.0875	29.1300	0.00000	21	1936.4079

Next, we considered 2 different random effects models – one a simple intercept only (mean) model, which seemed to be consistent with the strong support for the simple time variation model $\{\varphi_t\}$, and

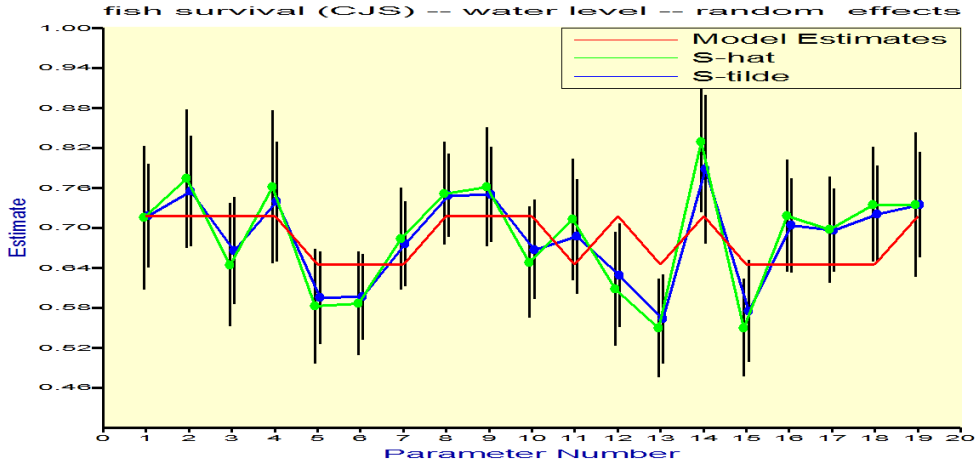
a model where survival was thought to vary randomly around a (water) level-specific mean. In other words, we hypothesized $\mu_{low} \neq \mu_{avg}$. In Appendix D, we also assumed that $\sigma^2_{low} = \sigma^2_{high}$. As noted, this is not directly testable using the moments-based variance components approach in **MARK**. But, in fact, estimating a separate μ and σ for each water level is possible, using an MCMC approach. (Recall that we included parameters $\varphi_1 \rightarrow \varphi_{19}$ in the random effect – φ_{20} was not included because it was confounded with p_{21}.) The results browser containing the 3 fixed effects models, and the 2 random effects models, is shown below:

Results Browser: Live Recaptures (CJS)

Model	AICc	Delta AICc	AICc Weight	No. Par.	Deviance
{phi(t)p(t): water level model -- Random Effects Trace G=15.2401619}	16546.4694	0.0000	0.58836	35.24016	1873.0840
{phi(t)p(t): intercept only model -- Random Effects Trace G=15.8100799}	16547.3766	0.9072	0.37380	35.81008	1872.8391
{phi(t)p(t)}	16551.9575	5.4881	0.03784	39	1870.9737
{phi(level)p(t)}	16570.9746	24.5052	0.00000	22	1924.2826
{phi(.)p(t)}	16581.0875	34.6181	0.00000	21	1936.4079

What is especially noteworthy here is that the random effects model with water level-specific means, $\{\varphi_{\mu_{level}\sigma^2_{level}}\}$, was the most parsimonious model in the model set, despite having no support whatsoever when considered in a fixed effects design. The near equivalence of the AIC_c weights between this model and the simpler 'intercept only' random effects model suggested that we couldn't differentiate between the two, but whereas our initial conclusion strongly rejected the hypothesis that there was an influence of water level on apparent survival, our random effects modeling would suggest that perhaps we shouldn't be quite so sure.

Here is a plot showing the ML and shrinkage estimates, and (importantly here) the underlying model (the red line), for model $\{\varphi_{\mu_{level}\sigma^2_{level}}\}$.



The red line clearly indicates that there were 2 separate means being modeled, for the low and average water flow years, respectively. The estimated process variance is $\hat{\sigma}^2 = 0.00313$, and the estimate of the estimate for $\hat{\beta}_1 = 0.0717$ in the linear model indicates that survival is higher in 'average' water level years (since we used 'low' level years as the reference level in our design matrix, above). What is also important here, is that the shrinkage estimates are clearly not constrained to fall exactly 'on the red line'

– they represent shrunk estimates of apparent survival as if each estimate was drawn randomly from a sample with a water level-specific mean.

Now, let's look at the estimates for μ and σ from the most parsimonious model, $\{\varphi_{\mu_{level}\sigma^2_{level}}\}$. The model was structured to estimate μ separately for each water level. But, recall because of how we structured the design matrix, what is estimated for μ is the linear model:

$$\text{logit}(\varphi) = \beta_1 + \beta_2(\texttt{level})$$

The estimates for the linear model coefficients were $\hat{\beta}_1 = 0.645584$, and $\hat{\beta}_2 = 0.071666$ (note that these coefficients are reported in the real scale, not the logit scale). Since we used a dummy coding of '1' for an 'average' water level year', then

$$\hat{\varphi}_{\text{avg}} = 0.645584 + 0.071666(1) = 0.717250$$

and

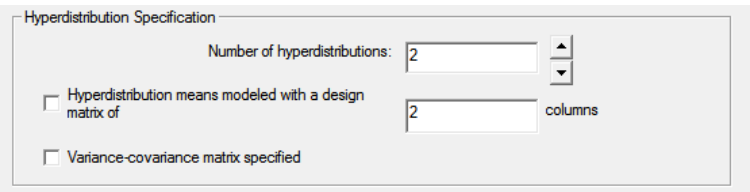
$$\hat{\varphi}_{\text{low}} = 0.645584 + 0.071666(0) = 0.645584$$

MARK also reports $\hat{\sigma} = 0.0559457$, but this is calculated over both water levels together (such that we might predict that 0.0559457 is the average of the process variances for each water level; we deal with this below). What we want, though, is an estimate of process variance for each water level separately. Is this possible using the 'method of moments' approach? The answer is 'no'. Both $\hat{\beta}_1$ and $\hat{\beta}_2$ are reported with an estimated SE, but these SE are estimated from the sum of process and sampling variation (i.e., total variance). Thus, even though we could, with a bit of algebra, use these SE to come up with estimates of the variance for both water levels, the calculated variance would be total variance for each water level, and not process variance, which is what we're really after.

Our solution is to use the MCMC capabilities in **MARK**, to directly estimate process variance for both water levels seperately. We simply specify 2 different hyperdistributions (one for average water level, and one for low water level), and estimate μ and σ for each hyperdistribution separately. Start by retrieving the fully time-dependent model $\{\varphi_t p_t\}$. We will re-run this model, using MCMC. We will specify the logit link, and will retrieve estimates from the fixed effects model $\{\varphi_t p_t\}$ to use as initial estimates.

Once you click the '**OK**' button, you're presented with the window which lets you set the MCMC parameters. We'll keep the default values for the number of samples, the specification of priors and will use a single chain. The only change we need to make to the defaults in this window is the number of hyperdistributions – now we want 2: one for average water level, and one for low water level.

So, the first thing we do is change the value in the box for '**Number of hyperdistributions**' from 0 to 2:

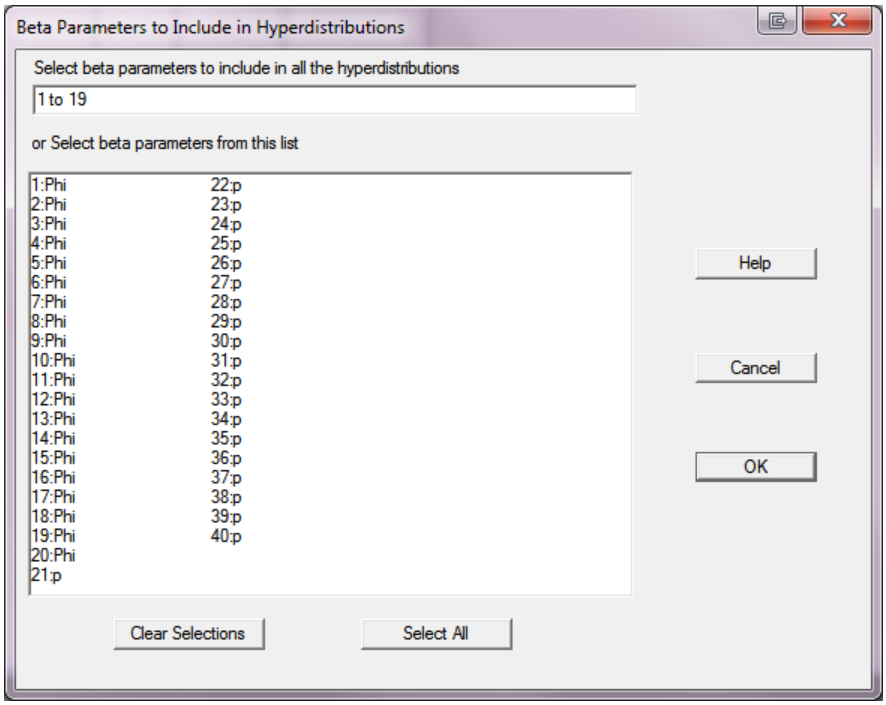


What we need to do next is specify which parameters are included in which hyperdistribution. With a bit of thought, you should realize that we want to include the parameters corresponding to average

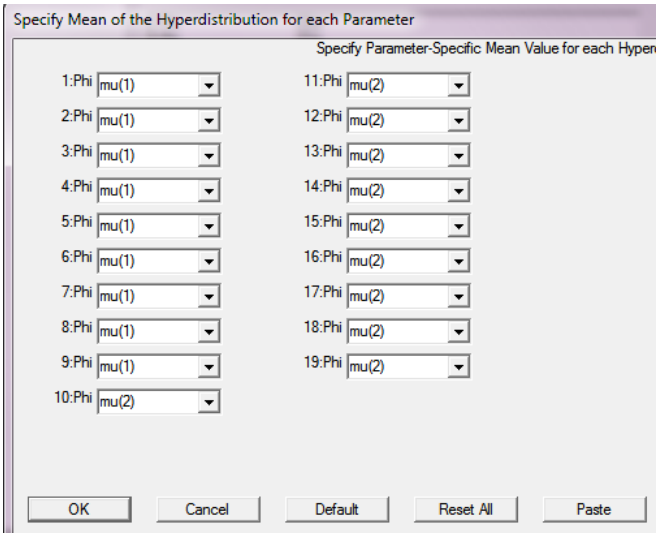
water years in one of the hyperdistributions, and the parameters corresponding to low water years in the other. **MARK** gives you a couple of ways to do this – in practice, you'll decide which one you find easier. Since this is the first time we've dealt with > 1 hyperdistribution, we'll demonstrate both approaches.

First, in the '**Hyperdistribution Specification**' box (preceding page), you'll notice that the second item in the box is a checkbox to allow you model the '**hyperdistribution means with a design matrix**'. This is turned 'off' (greyed out) by default. The fact that it doesn't immediately 'turn on' when you set the number of hyperdistributions > 1 suggests there is another, default approach to specifying the hyperdistributions.

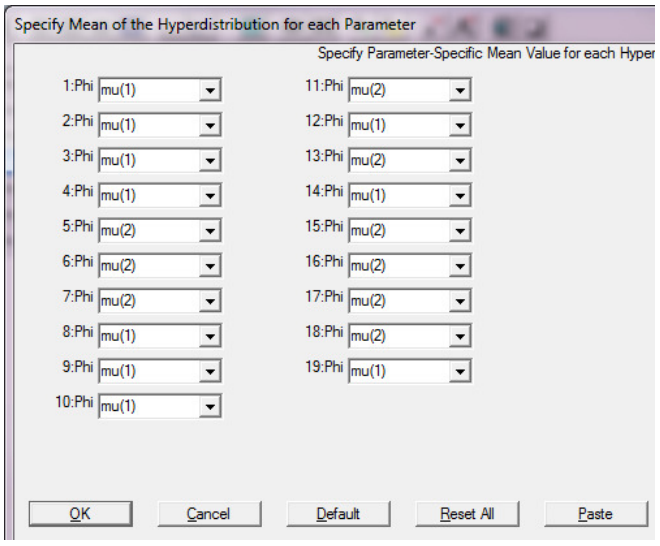
Indeed, there is. The default approach has a couple of steps. First, once you've set the number of hyperdistributions to 2 (above), click the '**OK**' button. This will bring up a window (shown below) where you list all the parameters which will be included in a hyperdistribution. You've seen this window before, but in the previous example, we had only a single hyperdistribution. Here, we're specifying 2 hyperdistributions. However, this is not the point where you assign a given parameter to a given hyperdistribution – it is merely where you tell **MARK** how many, and which parameters will, ultimately, be included in one hyperdistribution or another. For our purposes, we will enter '`1 to 19`':



Once we click the '**OK**' button, **MARK** will present you with a window (shown at the top of the next page) where you '**Specify the mean of the hyperdistribution for each parameter**'. By default (as shown), **MARK** will assume that if you specified two hyperdistributions, that half of the parameters will be included in the first hyperdistribution (i.e., μ_1), and the other half will be included in the second hyperdistribution (i.e., μ_2). But, this default might not be what you want. Moreover, if the total number of parameters over both hyperdistributions is odd (as it is in the present example, where we are including 19 parameters in the two hyperdistributions), you can't divide them evenly between the two hyperdistributions.

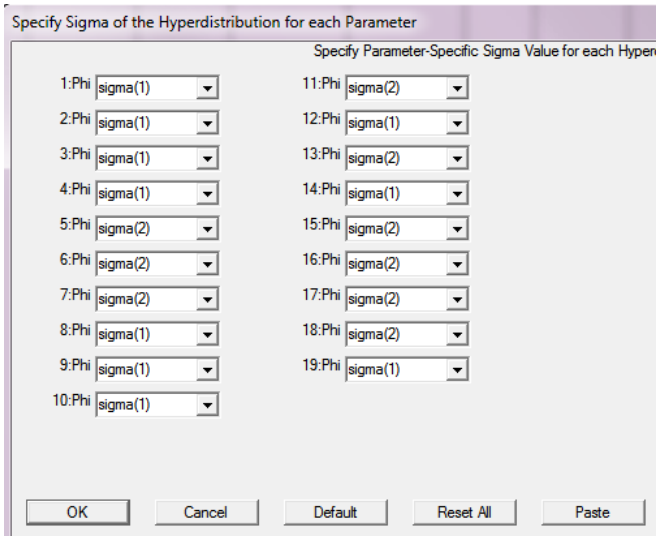


While this default is fine if a continuous sequence of parameters corresponds to a particular hyperdistribution, this is clearly not the case here, where each type of year (average water level, low water level) in our study is associated with a different hyperdistribution. Since this isn't what **MARK** defaults to, what you need to do at this point is go through the list of parameters – all 19 of them – and for each one in turn, decide which hyperdistribution they belong to, and select (from the drop-down menu) or manually enter the appropriate choice: `mu(1)` or `mu(2)`. Now, at this point, you need to remember which year (interval, parameter) corresponds to which water level. Recall that the time series of water level covariates was: {`AAAALLLAAALALALLLLAL`}, where '`A`' is average, and '`L`' is low. So, first 4 years had average water levels, followed by 3 years of low water levels, and so on. In the figure shown below, we've used `mu(1)` for average water level years, and `mu(2)` for low water level years. Note we only use the first 19 years, to eliminate confounding between the final φ and p estimates.



Once you click the '**OK**' button, **MARK** will present the exact same sort of window, except here you specify the σ hyperparameter for each hyperdistribution. We follow the exact same process as above,

assigning `sigma(1)` to average water years, and `sigma(2)` to low water years.



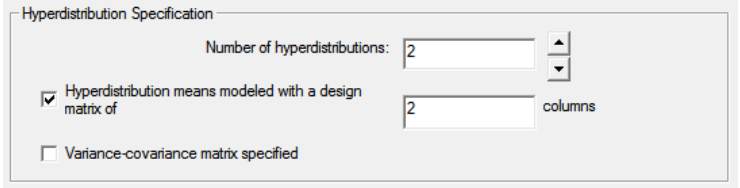
Once you click the '**OK**' button, you will be presented with the familiar window for specifying starting values, step size for the sampler, and the parameters governing the shape of the prior, for each of the 4 hyperparameters. For this example, we'll accept the defaults.

Finally, you're asked to enter initial parameter estimates, which you can retrieve from the appropriate model from the browser. Click '**OK**', the MCMC sampler will starting running.

Here are the estimates (on the logit scale) for `mu(1)` and `sigma(1)` (corresponding to $\hat{\mu}_{\text{avg}}$ and $\hat{\sigma}_{\text{avg}}$, respectively), and `mu(2)` and `sigma(2)` (corresponding to $\hat{\mu}_{\text{low}}$ and $\hat{\sigma}_{\text{low}}$), where '`avg`' and '`low`' are the two different levels of the water flow covariate.

```
39:p                    0.0051108       0.1313146       0.0031311       -0.0178870
40:p                    0.2805627       1.1353757       -0.1102470      -0.6830918
41:mu(1)                0.9493033       0.0930781       0.9468457       0.9192309
42:mu(2)                0.6155881       0.1199335       0.6143002       0.6284445
43:sigma(1)             0.2029072       0.1188329       0.1862616       0.1784712
44:sigma(2)             0.3101054       0.1218244       0.2879500       0.2621693
45:-2log Likelihood     16514.862       9.0845322       16514.222       16513.685
   -2log Likelihood for means of beta estimates =   16504.298
   DIC =   1922.8737
```

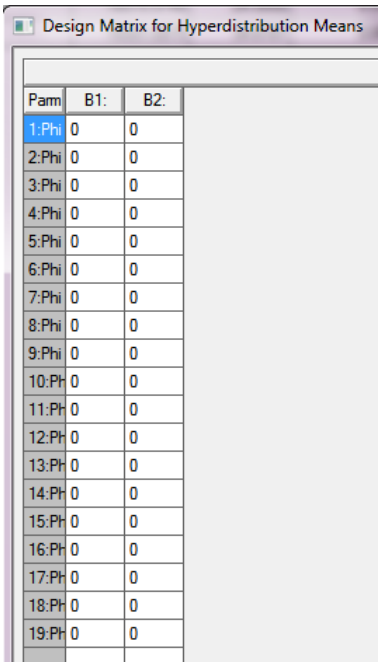
Now, before we back-transform the MCMC estimates from the logit scale to the real probability scale, let's first step back and look at the other way we could have specified the parameters in each hyperdistribution. We'll re-run our analysis, except that this time, after specifying 2 hyperdistributions, we'll go ahead and check the box which allows us to model the hyperdistribution means using a design matrix:



Once we check the box, we're asked to specify the number of columns in the DM. Since our classification factor has 2 levels (average, and low), you might instinctively (at this point in the book, answers to such questions concerning the DM might have reached the level of 'instinct') decide that you need 2 columns: 1 for the intercept, and 1 coding for water level. Now, as we'll see shortly, adopting this fairly standard (and familiar) linear models 'intercept offset' approach may not be the easiest or best approach to generating estimates of μ for each hyperdistribution.

After specifying 2 hyperdistributions, and indicating you want a DM with 2 columns, click the '**OK**' button. You'll then be asked to specify the parameters you want to include in the hyperdistributions ('`1 to 19`', for this example).

Now, when you click the '**OK**' button at this stage, **MARK** will generate a window that represents the familiar design matrix 'spreadsheet' (shown at the top of the next page), with the number of columns equal to what you specified a couple of steps ago (for this example, 2 columns). Notice that both columns are initially all 0's – meaning, you need to manually specify the structure you want for the means of the hyperdistributions.



Earlier, we suggested that most of you are probably imagining modifying this DM template to reflect a familiar 'intercept offset' linear model. In other words,

$$\text{logit}(\varphi) = \beta_1 + \beta_2(\texttt{water level})$$

where β_2 would represent the effect of changing water level relative to the reference level. If, for example, we used a '1' to code for the average water level, then β_2 would represent the degree to which apparent survival in years with an average water level deviated from the years with low water level. Alternatively, rather than using an 'intercept offset' approach, you could simply code each water level in its own column – in other words, if say the first column represented average water level, then entering a '1' for each average year. In the second column (which now represents low water level), you'd enter a '1' for those years with low water levels.

Both forms of the DM are shown in Fig. (E.8), below:

Parm	B1:	B2:
1:Phi	1	1
2:Phi	1	1
3:Phi	1	1
4:Phi	1	1
5:Phi	1	0
6:Phi	1	0
7:Phi	1	0
8:Phi	1	1
9:Phi	1	1
10:Phi	1	1
11:Phi	1	0
12:Phi	1	1
13:Phi	1	0
14:Phi	1	1
15:Phi	1	0
16:Phi	1	0
17:Phi	1	0
18:Phi	1	0
19:Phi	1	1

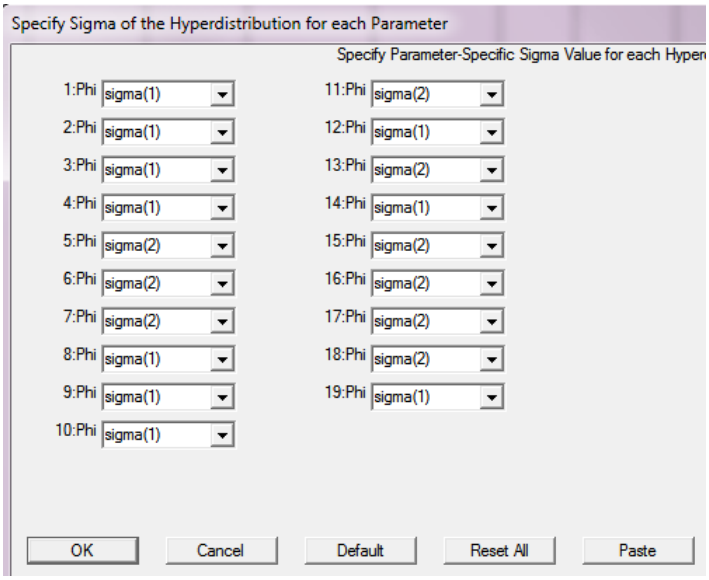
(a) standard 'intercept offset' coding

Parm	B1:	B2:
1:Phi	1	0
2:Phi	1	0
3:Phi	1	0
4:Phi	1	0
5:Phi	0	1
6:Phi	0	1
7:Phi	0	1
8:Phi	1	0
9:Phi	1	0
10:Phi	1	0
11:Phi	0	1
12:Phi	1	0
13:Phi	0	1
14:Phi	1	0
15:Phi	0	1
16:Phi	0	1
17:Phi	0	1
18:Phi	0	1
19:Phi	1	0

(b) single-factor 'identity' coding

Figure E.8: *Alternate DM structures for modeling water level effect*

In a moment, we'll address the question of which DM to use, and why. For the moment, assume we're using the 'intercept offset coding', shown in Fig. (E.8a). Once you click the '**OK**' button, you're presented with a window the lets you specify the σ for each hyperdistribution. In fact, it is exactly the same window we saw before for σ, and we modify it in exactly the same way:



So, we can use the DM to model the means of the hyperdistributions, but not the variances. This should make sense (since analysis of variance and linear models in general relate to structural relationships among means, and parsimonious estimates of those models, given the variance).

Go ahead and finish things up, and run the MCMC sampler for this model, coded using the 'intercept offset coding' DM. Here are results from our analysis:

```
39:p                    0.0086653       0.1285954       0.0088562       -0.0068439
40:p                    0.2756508       1.1628521       -0.1756704      -0.5326210
41:designbeta(1)        0.6150444       0.1092533       0.6092173       0.5804592
42:designbeta(2)        0.3278746       0.1450860       0.3290370       0.3822768
43:sigma(1)             0.1932216       0.1209980       0.1748662       0.0655464
44:sigma(2)             0.3022333       0.1132706       0.2818723       0.2603783
45:-2log Likelihood     16514.719       8.8534776       16514.144       16513.136
   -2log Likelihood for means of beta estimates =   16507.461
   DIC =   1919.4241
```

Now, re-do the analysis, but this time, use the alternate approach to coding the DM, shown in Fig. (E.8b). Here are the results from our analysis using that DM:

```
39:p                    0.0010649       0.1297505       0.0026155       -0.0012348
40:p                    0.4672519       1.0422928       0.2065664       -0.1892676
41:designbeta(1)        0.9514356       0.0931333       0.9461958       0.9310929
42:designbeta(2)        0.6114962       0.1139695       0.6117010       0.6072719
43:sigma(1)             0.1918787       0.1178243       0.1740883       0.1396514
44:sigma(2)             0.2999334       0.1094985       0.2840983       0.2723184
45:-2log Likelihood     16515.030       9.0251843       16514.489       16513.997
   -2log Likelihood for means of beta estimates =   16495.978
   DIC =   1931.5289
```

We quickly notice that the σ values are quite similar – this is not surprising, since the DM does not model σ.

For the estimates of μ, we'll start with the estimates derived from the model fit using the 'intercept offset coding' (Fig. E.8a). The `designbeta(n)` estimates correspond to the intercept (β_1) and slope (β_2) parameters in the following linear model:

$$\begin{aligned}\text{logit}(\hat{\varphi}) &= \hat{\beta}_1 + \hat{\beta}_2(\texttt{water level}) \\ &= 0.6150444 + 0.3278746(\texttt{water level})\end{aligned}$$

Thus, during an average water year,

$$\text{logit}(\hat{\varphi}) = 0.6150444 + 0.3278746(1) = 0.9429190$$

while in a low water year,

$$\text{logit}(\hat{\varphi}) = 0.6150444 + 0.3278746(0) = 0.6150444$$

Back-transforming from the logit scale to the probability scale for real parameters,

$$\begin{aligned}\hat{\mu}_{\text{avg}} &= \frac{e^{0.9429190}}{1 + e^{0.9429190}} & \hat{\mu}_{\text{low}} &= \frac{e^{0.6150444}}{1 + e^{0.6150444}} \\ &= 0.719689 & &= 0.649091\end{aligned}$$

which are close to the estimates from the 'method of moments' approach we derived earlier ($\hat{\mu}_{\text{avg}} = 0.717250$ and $\hat{\mu}_{\text{low}} = 0.645584$).

Now let's look at the estimates using the DM shown in Fig. (E.8b). We see that $\hat{\beta}_1 = 0.951436$, while $\hat{\beta}_2 = 0.611496$. What are these values? Simple – they are the estimates for $\text{logit}(\hat{\varphi}_{\text{avg}})$ and $\text{logit}(\hat{\varphi}_{\text{low}})$,

respectively! There is no linear model – they are estimates for each water level (average, low). [If you know why, good. If not, go back and study Chapter 6 again!] You'll notice that these estimates are similar to the values calculated on the logit scale for each water level, using the linear model (above). For example, 0.719689 vs. 0.717250 for the average water level. They should, in theory, be identical. And, if we'd used ML estimation on the data, they would be. But here we are using MCMC, and the slight differences between the values are because MCMC involves random sampling (and no 2 Markov chains will yield identical results). Both values are close to those derived by manually specifying the mean of the hyperdistribution for a set of parameters (top of p. E-28). In fact, this approach, and the approach using the DM shown in Fig (E.8b) are in fact entirely equivalent (again, the estimates reported differ slightly, due to the random nature of MCMC estimation).

Finally, what about process variation? Recall that a primary motivation for using MCMC for this example was because the MCMC approach allows us to estimate process variance σ separately for each water level. Using the Delta approximation, and our estimates from p. E-28, we find the back-transform of $\hat{\sigma}_{\text{MCMC, avg}} = 0.2029072$ on the logit to the real scale yields $\hat{\sigma}_{\text{avg}} = 0.040819$, while the back-transformation of $\hat{\sigma}_{\text{MCMC, low}} = 0.3101054$ on the logit to the real scale yields $\hat{\sigma}_{\text{low}} = 0.070622$. While we can't compare these values to estimates from the 'method of moments' approach (since that approach doesn't allow us to estimate σ separately for average and low water levels), we recall (from earlier in this section) that the overall estimated process variance for both water levels together from the 'method of moments' analysis was $\hat{\sigma} = 0.0559457$, which is quite close to the average of the estimates of σ we just derived for each water level using MCMC ([0.04082+0.07062]/2=0.05572).

Some closing comments...

In this example, we clearly didn't need to use a design matrix (DM) to achieve our objective. Our intent was simply to demonstrate some of the mechanics, rather than suggesting this was a good approach for this particular problem. As you know from other chapters in this book, the DM is a powerful tool for modeling parameters. The same applies here, but keep in mind that the DM available in the MCMC part of **MARK** are more limited than the full-blown DM you've used for linear models for various data types (in particular, you are using the DM to model the means of parameters, and the number of means will always be less than the number of occasions).

Also, you may have noticed that we've progressed fairly far into this Appendix without making much reference to 'model selection'. While we did mention earlier that **MARK** generates the DIC for a given model, we did not do much more than define it, and suggested that you're welcome to use it, but you're on your own if you do (there is a fair bit of literature out concerning use of the DIC for model selection – especially since the DIC is automatically generated by software like **WinBUGS** or **OpenBUGS**). This isn't because we've suddenly decided that model selection (and related things like model averaging) are any less important. On the contrary, these continue to be vitally important for the way we analyze data to address interesting questions. Unfortunately, model selection in the Bayesian context (where you generally find MCMC being used most frequently (pun completely intended...) is not as straightforward as evaluating models based on nominal AIC weights. A lot of smart folks are thinking pretty hard about this problem.

E.2. Hyperdistributions between structural parameters

In the second motivating example presented at the start of this appendix ('scenario 2'; p. E-3), we considered an example based on analysis of dead recovery data, where we had interest in the correlation between the two structural parameters S (survival) and f (recovery rate). This interest was motivated by some *a priori* expectation that predicted that the sign of any correlation between S and f might

indicate support for one of two competing models relating harvest to mortality. As noted, the difficulty in assessing the correlation between these two parameters is that there is covariance both within (over sampling occasions) and between the two parameters. Thus, the parameters are not independent samples, and we can't simply take estimates of $\hat{S}_i$ and $\hat{f}_i$ and estimate the correlation. This would amount to little more than 'doing statistics on statistics', which is almost always a poor approach.

Here we consider 2 worked examples – one based on a dead recovery analysis, and the other on a Pradel model approach. Our intent is to focus on the mechanics, rather than provide an exhaustive list of models where this approach could be implemented. But, it may suffice to say that the approach we'll describe here could be used to model covariance among structural parameters in any model where there might be interest. Other than greater flexibility for handling estimation of process variation (which has been our focus up until this point), the ability to consider multivariate hyperdistributions is one of the main motivations for use of MCMC in **MARK**. Not only are the types of questions which can be addressed using this approach numerous and varied, in most cases, using MCMC is the only viable means of considering them.

E.2.1. Example 1 - dead recovery analysis (corr{S, f})

For this example we consider the estimation of the correlation of 2 structural parameters in a Brownie dead recovery model (survival, S, and recovery rate, f). To demonstrate the mechanics of using MCMC in **MARK** to estimate the correlation, we simulated a data set with 11 occasions (thus, 10 estimates of survival, and 11 estimates of recovery rate). We drew our set of underlying survival and recovery probabilities used to simulate the data from a random bivariate normal distribution with $S_i = \mathcal{N}(0.8, 0.025)$, and $f_i = \mathcal{N}(0.1, 0.015)$, with $\rho(S, f) = -0.65$. In other words, we drew a random sample of survival and recovery probabilities from a distribution where the 2 parameters were negatively correlated. Recall (from Chapter 8) that there is one more recovery parameter ($k = 11$) than survival parameter ($k = 10$). Thus, we drew 10 pairs of parameter values for $S_{1\rightarrow 10}$ and $f_{1\rightarrow 10}$. For f_{11}, we used $f_{11} = 0.1$, which was the true mean of the distribution from which the f_i were drawn. Here is the sample of probabilities we ended up with for each parameter (rounded to 3 digits).

survival (S)	recovery (f)
0.804	0.118
0.802	0.099
0.807	0.097
0.790	0.109
0.846	0.085
0.787	0.114
0.818	0.092
0.798	0.107
0.797	0.097
0.793	0.123
–	0.100

The correlation of these true parameters used in our simulation $\rho(S_{1\rightarrow 10}, f_{1\rightarrow 10}) = -0.76$ (remember – we took a small sample of parameter values from the underlying bivariate normal distribution with overall $\rho = -0.65$. The value -0.76 is the estimate of ρ from this sample). It is this correlation between $\rho(S_{1\rightarrow 10}, f_{1\rightarrow 10})$ that we will try to estimate using MCMC.

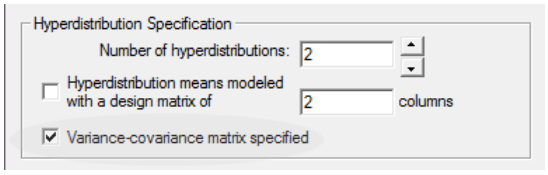
For each sampling occasion, we simulated marking and release of 10,000 individuals.* We ran a single simulation, and used the encounter histories generated by that simulation for our analysis. These simulated encounter data (in LDLD format) are contained in `brownie-corr.inp`.

Start **MARK**, select '**Dead recoveries | Brownie**' as the data type, set the number of occasions to 11. Click '**OK**', and then go ahead and fit the full time-dependent default model, $\{S_t f_t\}$, using the logit link. Add the results to the browser.

Now, we're going to re-run this model, using MCMC. Run the model, but this time, check the '**MCMC estimation**' check-box. It is also a good habit to use initial parameter estimates to start the sampler, so check the '**Provide initial parameter estimates**' box as well. Then click '**OK**'.

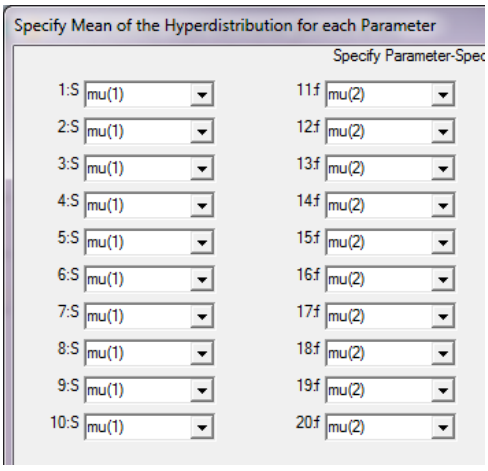
Next, we need to set various parameters to govern the MCMC sampler. We'll use the default 4,000 tuning and 1,000 burn-in samples, but we'll increase the number of samples to store from 10,000 to 50,000 (you'll see why in a moment).

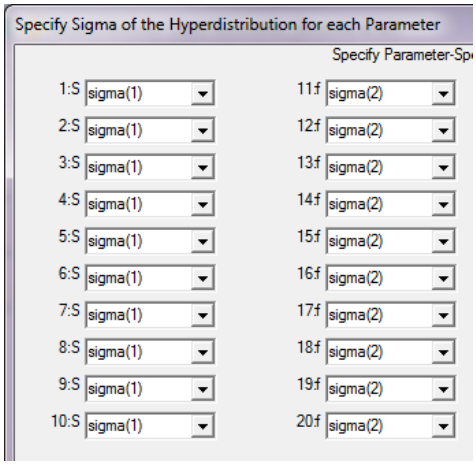
Next, we need to specify 2 hyperdistributions (one for S, and one for f). And, now, the new step – we need to check the box to allow us to specify the '**Variance-covariance matrix**', as shown in the following:



After you click '**OK**', we're asked to specify the '**Beta parameters to include in the hyperdistributions**'. Our interest is in the correlation between $S_{1\rightarrow10}$ and $f_{1\rightarrow10}$, so we enter '`1 to 10, 11 to 20`'.

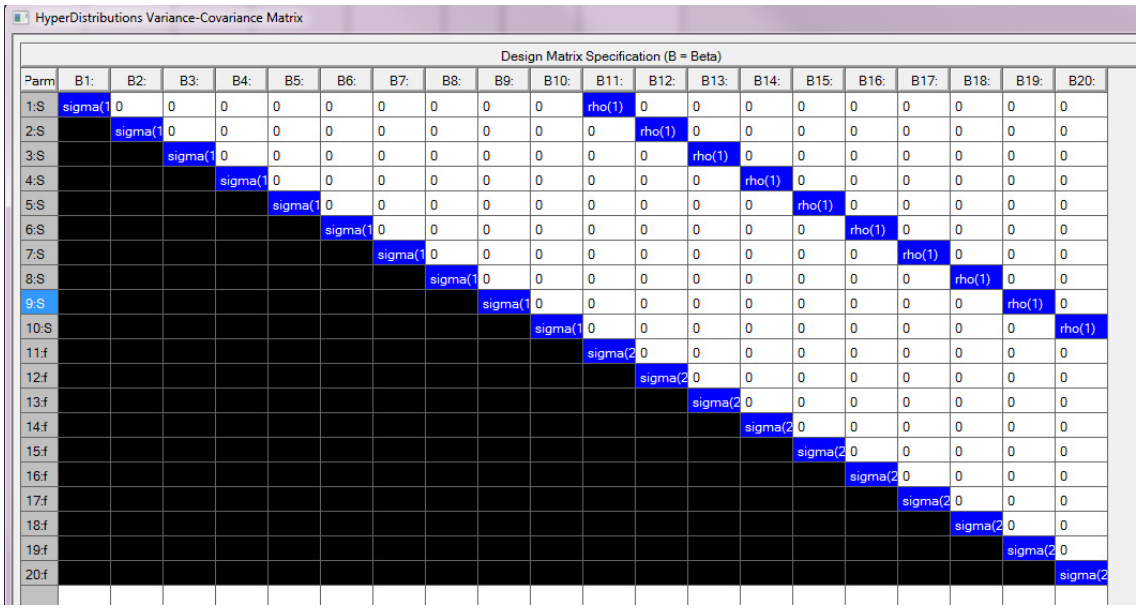
In the next step, **MARK** defaults to precisely what we want – the mean (`mu(1)` and `mu(2)`) and variance (`sigma(1)` and `sigma(2)`) are correctly associated with parameters $1 \rightarrow 10$ and $11 \rightarrow 20$, respectively. Remember, there is no S_{11} corresponding to f_{11}, so we use only 10 for each parameter.





Finally, we're at the point where we want to specify the variance-covariance structure for the 2 hyperdistributions.

* In fact, for many important harvested species, samples of newly marked and released individuals are often as larger, or larger.



Parm	B1:	B2:	B3:	B4:	B5:	B6:	B7:	B8:	B9:	B10:	B11:	B12:	B13:	B14:	B15:	B16:	B17:	B18:	B19:	B20:
1:S	sigma(1	0	0	0	0	0	0	0	0	0	rho(1)	0	0	0	0	0	0	0	0	0
2:S		sigma(1	0	0	0	0	0	0	0	0	0	rho(1)	0	0	0	0	0	0	0	0
3:S			sigma(1	0	0	0	0	0	0	0	0	0	rho(1)	0	0	0	0	0	0	0
4:S				sigma(1	0	0	0	0	0	0	0	0	0	rho(1)	0	0	0	0	0	0
5:S					sigma(1	0	0	0	0	0	0	0	0	0	rho(1)	0	0	0	0	0
6:S						sigma(1	0	0	0	0	0	0	0	0	0	rho(1)	0	0	0	0
7:S							sigma(1	0	0	0	0	0	0	0	0	0	rho(1)	0	0	0
8:S								sigma(1	0	0	0	0	0	0	0	0	0	rho(1)	0	0
9:S									sigma(1	0	0	0	0	0	0	0	0	0	rho(1)	0
10:S										sigma(1	0	0	0	0	0	0	0	0	0	rho(1)
11:f											sigma(2	0	0	0	0	0	0	0	0	0
12:f												sigma(2	0	0	0	0	0	0	0	0
13:f													sigma(2	0	0	0	0	0	0	0
14:f														sigma(2	0	0	0	0	0	0
15:f															sigma(2	0	0	0	0	0
16:f																sigma(2	0	0	0	0
17:f																	sigma(2	0	0	0
18:f																		sigma(2	0	0
19:f																			sigma(2	0
20:f																				sigma(2

What you see pictured above is the variance-covariance matrix, for the 2 hyperdistributions. Along the left-hand size, you'll see a column listing the parameters $S_1 \rightarrow S_{10}$, and $f_1 \rightarrow f_{10}$. There is no horizontal reference bar analogous to the vertical reference bar in the linear models DM separating different types of parameters. Along the diagonal in blue (i.e., immediately above the 'lower-triangle' in black) we see 20 'blue boxes' with white lettering (actually, unless you've changed the default DM colors in **MARK**, what you will see on your computer is red boxes with white lettering – for purposes of increasing contrast here in the book, we've changed from a red color scheme to blue). Starting from the upper-left corner, and moving down the diagonal, the first 10 cells along the diagonal are labeled '`sigma(1)`', while the next 10 are labeled '`sigma(2)`'. Remember, in a variance-covariance matrix, the variances of the parameters are on the diagonal. That is exactly what we see here. But, because we have 'collected' parameters into hyperdistributions, we see '`sigma(1)`' for parameters $1 \rightarrow 10$, rather than '`sigma(1)`', '`sigma(2)`',. . . ,'`sigma(10)`'.

Now, you should also recall that off the diagonal in a V-C matrix are the covariances between particular parameters. These you will need to manually enter. If you leave the above diagonal cells at the default of '`0`', you would end up simply estimate process variance, '`sigma(1)`' and '`sigma(2)`'. However, recall what we are trying to do here – we want to estimate the correlation, ρ, between $S_{1\rightarrow10}$ and $f_{1\rightarrow10}$, which correspond to parameters $1 \rightarrow 10$ and $11 \rightarrow 20$, respectively. So, S_1 is paired with f_1, S_2 is paired with f_2, and so on. Meaning, we need to find the point above the diagonal where parameter S_1 (row 1), pairs up with parameter f_1 (row 11 – note that the row numbers match the parameter indexing). So in matrix element [1, 11], we manually type in – carefully – the expression '`rho(1)`'. We do the same thing for parameter S_2 (row 2), and corresponding recovery parameter f_2 (row 12) – in matrix element [2, 12] we again enter the expression '`rho(1)`'. We do not enter '`rho(2)`' (i.e., we don't increment the indexing for '`rho(`n`)`'), because we are estimating one correlation coefficient (i.e., '`rho(1)`') between both sets (hyperdistributions) of parameters.

So, off the diagonal, a diagonal vector of 10 elements where you've entered the expression '`rho(1)`'. You can do this manually, or it may be more efficient to use the design matrix command '**`Copy Value Diagonal`**', especially for large matrices where many values of the rho parameter must be entered. You also have the option to paste a VC matrix into the window, but you may want to specify the entire matrix (both above and below the diagonal) and paste it into the window, because trying to construct

the values to paste with different numbers of elements per row can be painful. For the values below the diagonal, specify zeros, as these will be ignored.

Note that only zeros or '`rho`' values are allowable values in the upper off-diagonal portion of the matrix, and only `sigma(n)`'s are valid on the diagonal. So, technically, this matrix is a correlation matrix *above* the diagonal, and a standard deviation vector *on* the diagonal.

begin sidebar

more on the MCMC VC matrix...

As mentioned, the default VC matrix is all zero values, which is the same as if no variance-covariance matrix is specified. On the diagonal of the matrix are the sigma values (which you should not need to change if they were correctly specified in the specification of the hyperdistribution). You only have to specify the off-diagonal elements above the diagonal, as the matrix is symmetric and the lower off-diagonal elements should be blacked out.

In the present example, we've specified a VC matrix to estimate the correlation, `rho(1)`, between 2 hyperdistributions. Another potentially useful matrix to estimate the *autocorrelation* of the beta parameters is the following (we show the first 5 beta parameters only):

`sigma(1)`	`rho(1)`	`rho(1)**2`	`rho(1)**3`	`rho(1)**4`
	`sigma(1)`	`rho(1)`	`rho(1)**2`	`rho(1)**3`
		`sigma(1)`	`rho(1)`	`rho(1)**2`
			`sigma(1)`	`rho(1)`
				`sigma(1)`

This VC matrix models the correlation between *consecutive* parameters. If β_1 and β_2 are correlated as `rho(1)`, and β_2 and β_3 are also correlated as '`rho(1)`', then β_1 and β_3 have to be correlated with '`rho(1)**2`'. This autocorrelation matrix can be obtained easily by right-clicking the VC matrix window and selecting the '**`Autocorrelation matrix`**' option for the list of options. Likewise, you can get back to the default matrix of zero off-diagonal elements with the '**`No correlation`**' menu choice.

end sidebar

Back to our example - Once you've completed filling out the VC matrix, click the '**`OK`**' button.

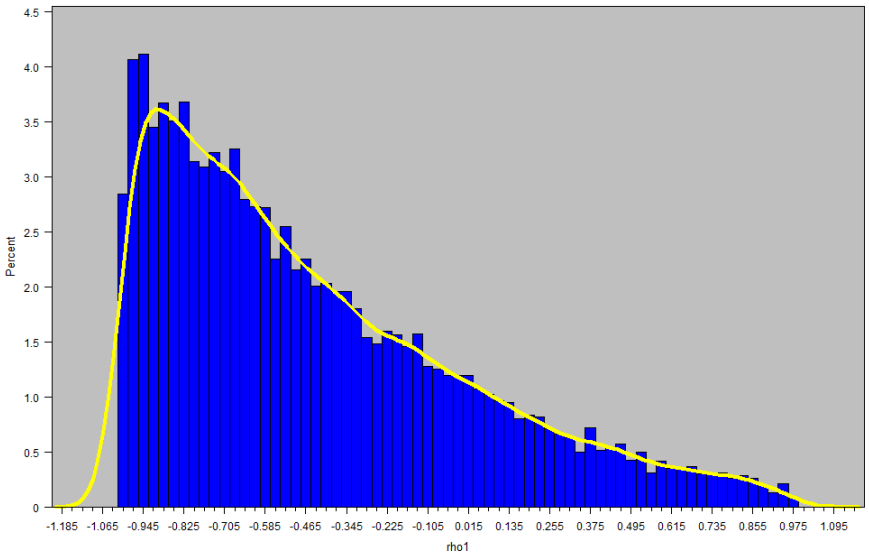
We'll accept the defaults for the priors – but will note that the prior for the correlation ρ is $\mathcal{U}(-1, 1)$. More on this in a moment.

Once you've pulled in the initial parameter estimates from the appropriate model in the browser, go ahead and run the MCMC sampler. Here are the results we generated, based on 50,000 samples:

```
20:f                    -1.9664312      0.0179214       -1.9661824      -1.9641209
21:f                    -2.2156381      0.0211403       -2.2158994      -2.2173419
22:mu(1)                1.4241278       0.0263609       1.4247749       1.4275046
23:mu(2)                -2.1617753      0.0506300       -2.1614787      -2.1560266
24:sigma(1)             0.0631657       0.0329045       0.0560224       0.0455499
25:sigma(2)             0.1551130       0.0425683       0.1471698       0.1334634
26:rho(1)               -0.4433035      0.4553181       -0.5606491      -0.9999635
27:-2log Likelihood     268155.78       5.3284515       268155.19       268154.08
   -2log Likelihood for means of beta estimates =   268141.63
   DIC =   78.075570
```

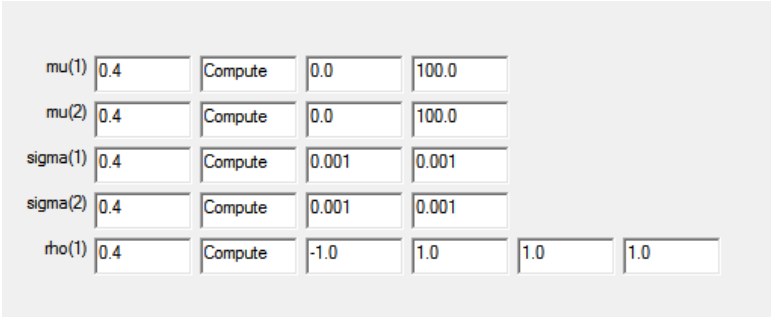
Note that the estimate of the correlation – based on correlation on the logit scale – is $\hat{\rho} = -0.443304$. We need to pause and make several points here.

1. Earlier, we noted that we were going to increase the number of samples from the default of 10,000 to 50,000. Why? Well, to answer, let's have a look at the frequency histogram (top of the next page) based on the distribution of 50,000 samples from the posterior for ρ.



Clearly, this is not a nicely symmetrical distribution – as such, the most appropriate 'moment' of this distribution to use for inference about a point estimate for ρ is open to some debate. From the results shown above, we see estimates of -0.443304, -0.56065 and -0.99996 for the mean, median, and mode, respectively. Clearly, the reported mode isn't plausible. But, what about the mean and median values? Which one should we choose? In this instance, the point may be moot since the 95% frequency-based credibility interval for $\hat{\rho}$ is $[-0.9917, 0.6739785]$, which clearly bounds 1.0. So does the 95% HPD, $[-1.000, 0.2729]$. This is not particularly encouraging, given large samples of marked and released individuals in the simulation, and a relatively strong true negative correlation between S and f simulated in the data. Some solace, perhaps, given that the estimate is at least in the right direction (negative), although we know this only because we know the underlying structure of the true model we used to generate the data. Keeping all of this in mind, consider that the situation would have been even more uncertain if we'd used the default of 10,000 samples, instead of 50,000 (try it for yourself).

2. How would the situation change if we used *a priori* expectation, and changed the prior on ρ from $\mathcal{U}(-1, 1)$ to $\mathcal{U}(-1, 0)$, or perhaps a different, informative prior (say,$\mathcal{B}(5, 2)$)? You might recall that when we ran our analysis, the last step involved specifying the priors for the univariate and any multivariate hyperparameters.



For the univariate hyperparameters μ_i and σ_i, four inputs are requested. The first edit

box is to specify the step size to be used to generate new parameter values with which to sample the posterior distribution. As with the default step size, the goal is to tune the estimation so that approximately 45% of the steps are accepted. The second edit box is to specify an initial value to start the Markov Chain. The default is '`compute`', which tells **MARK** to compute an estimate from the initial values of the beta parameters. It is generally recommended practice that you should run the model that you want to use for MCMC estimation as a typical **MARK** analysis, so that you can provide initial estimates to start the MCMC estimation.

The third and fourth edit boxes specify the parameters for the *prior* distribution to be used with this hyperdistribution. For mean parameters, a normally distributed prior is used. The third edit box specifies the mean, and the fourth edit box specifies the standard deviation. The default values for μ are mean = 0 and standard deviation = 100, giving a very flat and uninformative prior over the range of the possible values of the mean. For σ parameters, an inverse gamma distribution prior is used with parameters alpha and beta. The mean of a gamma distribution is α times β, and the variance is α times β^2. Values of $\alpha = 0.001$ and $\beta = 0.001$ result in a reasonably flat prior distribution. In other words, the defaults specify uninformative 'flat' priors for μ and σ.

For the multivariate hyperparameter, ρ, the default prior is a beta distribution over the range specified by the lower bound in the third edit box to the upper bound in the fourth edit box (see figure at bottom of preceding page). The default values for the range of ρ are $[-1, 1]$, but to force the correlation to be positive, you might use $[0, 1]$, or forced to be negative, $[-1, 0]$.

In addition, the fifth and sixth boxes for ρ (above) specify the α and β parameters of the beta distribution (see - `sidebar` - starting on p. E-15). The default is $\alpha = \beta = 1.0$, which gives a uniform distribution on the range specified (see Fig. E.4 on p. E-16). The shape of the beta prior on ρ over the interval $[-1, 1]$ can be changed by modifying the values of α and β (Fig. E.10):

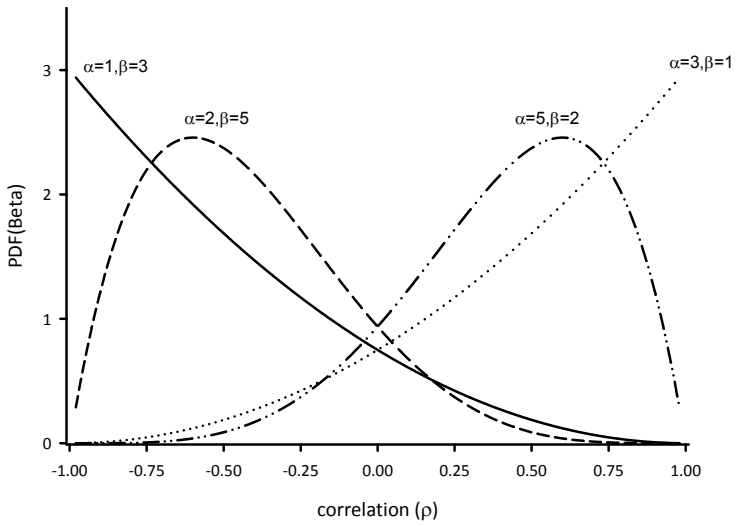


Figure E.10: *Shape of the beta probability distribution for unequal values of the shape parameters α and β. The distribution is symmetric around $\rho = 0$ over the interval $[-1, 1]$ for $\alpha = \beta$.*

For our present example, lets see what happens if we use a beta prior for ρ which gives greater weight to negative values (using, say, $\mathcal{B}(2,5)$ Doing so changes the estimates of the mean and median on the logit scale for ρ to -0.5389 and -0.5863, respectively. Both are negative. But, what about the credibility intervals? The frequency percentile-based 95% interval is $[-0.9298, 0.1105]$, while the HPD is $[-0.1275, 0.0046]$. We might be somewhat comforted by the fact that either credibility bound is 'more supportive' of a negative correlation. Of course, the burden would now fall on us to justify the use of an informative prior, but that is part of the 'fun' of 'being Bayesian' (at least by some definitions).

3. However, while we might be increasingly satisfied/content with our results, at least those based on the use of the informative beta prior $\mathcal{B}(2,5)$ (see point 2, above), we must think – hard – about whether or not this estimate of ρ is robust, and whether or not it represents what we think (hope?) it does – the covariation of survival S and recovery rate f. In fact, there might be a real problem here – since $f = Kc\lambda$ (discussed earlier in the appendix), then in fact there is an implicit relationship between survival S and the probability of mortality due to harvest, which is clearly a function of f. In fact, under some assumptions, there is an intrinsic 'part-whole' correlation between S and f such that the null hypothesis for our analysis might not be $\rho = 0$, but rather, the null expectation for ρ might be some other non-zero value. As such, a fair and robust interpretation of $\hat{\rho}$ would depend on knowing what this intrinsic correlation might be (for a brief discussion of one approach to estimating the covariance function for S and f, see last example in Appendix B).

4. Finally, the estimate $\hat{\rho}$ is reported on the logit scale. OK – you might think, no problem. Back-transform the estimate from the logit scale to the real probability scale. But, take a moment to think about it. The correlation ρ is estimated on the interval $[-1, 1]$. A correlation of $\rho = 0.0$ (i.e., no correlation) would back-transform to $\exp(0)/(1 + \exp(0)) = 0.5$. If the correlation was perfectly negative, $\rho = -1.0$, then the naive back-transformed value would be $\exp(-1)/(1+\exp(-1)) = 0.269$. In fact, a parameter bounded on the interval $[-1, 1]$ on the logit scale would back-transform to a parameter bounded on the interval $[0.269, 0.731]$ on the real probability scale, not $[-1, 1]$ or even $[0, 1]$ – the latter would require ρ being estimated on the logit scale on the interval $[-\infty, +\infty]$, while the former is not mathematically possible, since the back-transform from the logit cannot generate a transformed value < 0.

   Fortunately, it doesn't matter here. Imagine you have some estimates of 2 parameters on the real probability scale, say S and f. Suppose they have a true correlation on the real scale of ρ_{real}. It turns out that if you take the data (i.e., the parameter estimates on the real scale), and logit transform them, and then calculate the bivariate correlation on the logit transformed data, you'll find that $\rho_{\text{logit}} \simeq \rho_{\text{real}}$. You can easily prove this for yourself – simulate a number of sets of parameters from a bivariate normal with known correlation, transform the data from the real scale to the logit scale, calculate the correlation of the logit transformed data, and compare with the correlation on the real scale. Given a large enough number of simulations, you'll see that indeed $\rho_{\text{logit}} \simeq \rho_{\text{real}}$.

   Why? Simple – because you're estimating a parameter (ρ) describing the linear covariance between two parameters, and the transformation from real to logit is itself effectively linear, over the typical range of data. It is a 'basic result' that the bivariate correlation of X_1 and X_2 is invariant to a linear transformation. It is also *largely* invariant to nonlinear transformation, provided the transformation is *monotonic* (in other words, the sin link would not work particularly well). Again, this is easy enough to demonstrate for yourself. In other words, saying that '$\hat{\rho} = xyz$ on the logit scale' is effectively the same as saying that '$\hat{\rho} = xyz$ on the familiar $[-1, 1]$ scale'.

An additional way you can confirm this for yourself is to re-run the MCMC analysis, using the identity link. Normally we don't recommend using the identity link function, but it tends to be fairly stable for Brownie dead recovery models. The practical advantage of using the identity link is there is no extra 'thinking' involved in considering whether a parameter is estimated on this scale or that, or how to transform to the real probability scale. Based on 50,000 samples, and using the identity link, the mean and median for the estimated posterior for $\hat{\rho}$ were -0.60791 and -0.80254, respectively. Since the logit transform is essentially linear here, we would not anticipate that using an identity link would change the estimates much (they don't - both the mean and median are relatively close to what we saw above using the logit link).

5. So, we conclude that our best estimate of the correlation between S and f for our simulated data is probably $\hat{\rho} \approx (-0.5 \leftrightarrow -0.65)$. But, the credibility interval nearly bounds 0 (using an informative beta prior on ρ), so we wouldn't be all that excited by this result. In fact, we should probbly be suspicious of trying to 'tell a story' based on only 10 pairs of parameters in the first place. We re-visit the issue of the length of the time series used in specifying the hyperdistributions later.

E.2.2. Example 2 - Pradel model

A final example application considers correlation between the process parameters φ (apparent survival) and f (recruitment), in a Pradel temporal symmetry model (Chapter 13). We will re-visit the moth (*Gonodontis bidenta*) data analysis introduced in Appendix D. The data (contained in `moth-example.inp`) consist of records for 689 male moths that were captured, marked, and released daily over 17 days in northwest England. These moths were nonmelanic; demographic parameters were estimated as part of a larger study looking at comparative fitness of distinct color morphs.

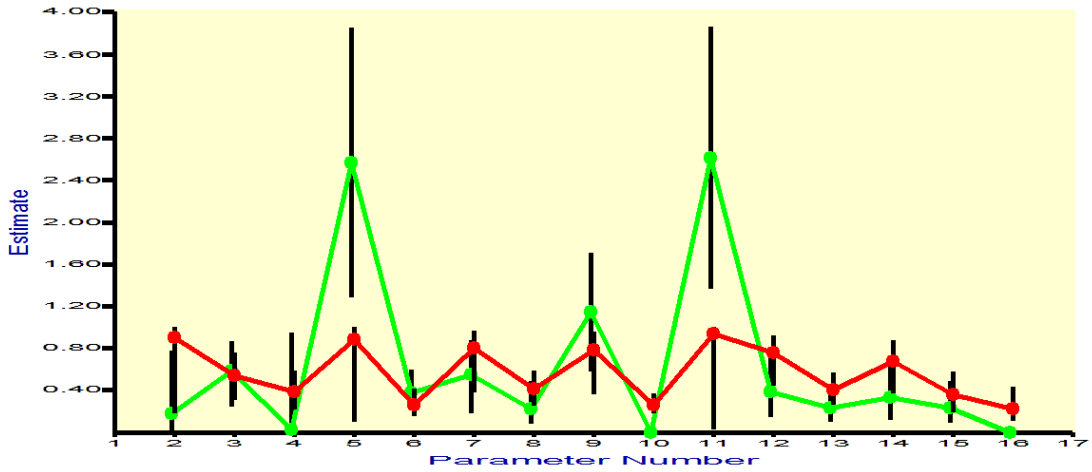
In Appendix D, we used 'method of moments' random effect models, focussing on estimation of process variance, and possible trend, in realized growth rate λ. Our focus here is on estimating correlation between the process parameters which jointly determine realized population change (since $\lambda = \varphi + f$). We might be interested if they covary positively (such that high survival years are also characterized by high recruitment – the 'a good year for one parameter is a good year for another' hypothesis), or negatively (such that increased recruitment, say, correlates with reduced survival. Potentially evidence for the ubiquitous 'density-dependence' hypothesis which has featured prominently in population ecology for a long time).

Go ahead and start **MARK**, and import the moth data. Select the '**`Pradel survival and recruitment`**' data type with 17 sampling occasions. For purposes of convenience, let's set the starting general model to $\{\varphi_t p_{\cdot} f_t\}$. By fixing encounter probability p to be constant over time, we're eliminating some of the potential confounding between φ and f in a fully time-dependent model. The key word here is 'some'. Even with a time-constant p, experience has shown that the first estimates of λ (and as such, φ and f) are likely biased. So, we'll exclude the φ_1 and f_1 from our hyperdistributions, and base our MCMC estimation of correlation between φ and f on the remaining 15 estimates.

We'll fit our general model using '**`parm-specific`**' link functions. For φ and p, both of which are bounded $[0, 1]$, we'll use the logit link. For f (recruitment), which cannot be negative (lower bound of 0), but which can be (and frequently is) ≥ 1, we'll use the log link. We'll also use simulated annealing for the optimization (preliminary analysis shows that the estimation of some parameters for these data is somewhat sensitive to starting values used in the optimization).

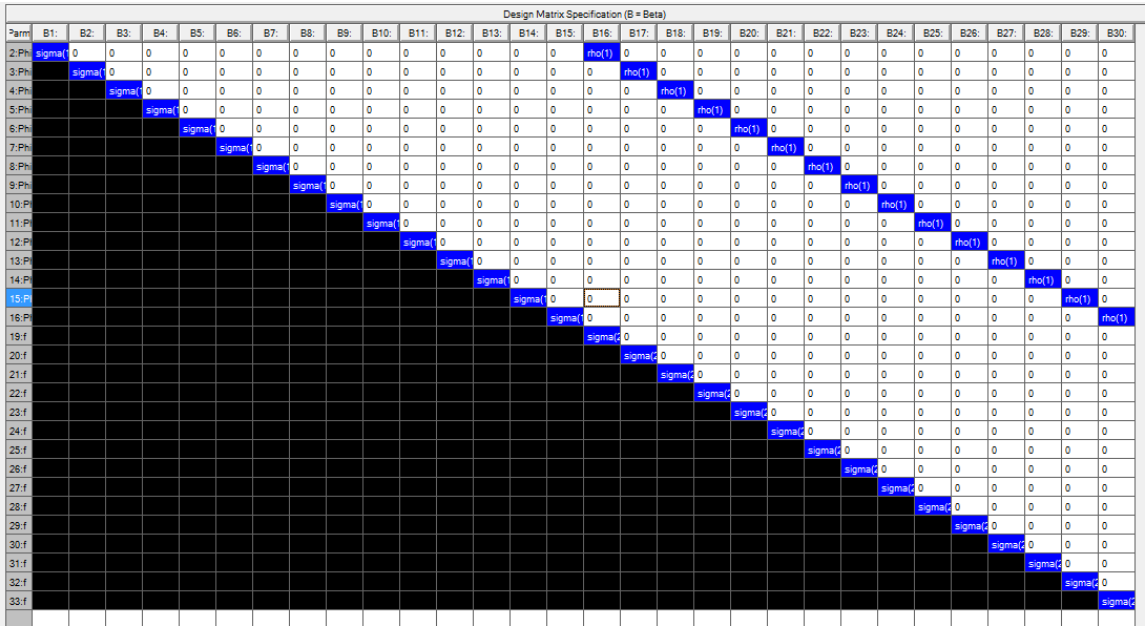
[*Note*: using '**`parm-specific`**' link functions, or the log link for all parameters, yields the same model deviance. However, the parameter count differs depending on which link function approach is used.]

Here is a plot of $\hat{\varphi}_i$ and $\hat{f}_i$, for occasions $i = 2$ to 17, where the red line represents estimates for apparent survival $\hat{\varphi}_i$, and the green line represents estimates for per capita recruitment, $\hat{f}_i$.



There would seem to be some evidence – based on the completely non-rigorous criterion of 'visual assessment' of positive covariation between the two parameters. We of course prefer a better approach, and will proceed with a formal MCMC estimation of the correlation between φ_i and f_i.

Proceed as in the previous example – let's set the number of samples to 50,000. For specifying the hyperdistributions, we want to use parameters 2 to 16 for one, and 19 to 33 for the other (remember, we are dropping the first φ and f estimates from the hyperdistributions). Once you have set up `mu(1)` and `mu(2)`, and `sigma(1)` and `sigma(2)`, you'll be presented with the template for the variance-covariance matrix. Here, because we have 30 total parameters, the VC matrix is getting a bit dense (shown below) – almost rivaling some of the DM seen for the more complicated robust design models introduced in Chapter 15.



However, if you understood what we did in the preceding example, then this one should not be much more difficult, despite the scale of the VC matrix. You simply need to remember to 'match' up pairs of φ and f parameters. In this case, φ_2 with f_2, φ_3 with f_3, and so forth. The parameter φ_2 is in row 1, while parameter f_2 is in row 16. This, we click in the cell in row 1, column 16, and enter '`rho(1)`'. We then simply copy this down the diagonal (using the right-click menu design menu option), as shown. With some practice, this tends to be relatively easy to accomplish.

Proceed through the next steps, retrieving estimates from the appropriate model in the browser as initial values, and then start the sampler. Here are the results from our analysis, based on 50,000 samples.

```
32:f                    -1.8793636      0.5398714       -1.8139282      -1.6532947
33:f                    -3.6299603      0.9897197       -3.4812543      -3.2572262
34:mu(1)                0.3479257       0.3453461       0.3160946       0.2813022
35:mu(2)                -1.2506498      0.4283580       -1.2128292      -1.1596338
36:sigma(1)             1.0592197       0.3652815       0.9969333       0.8981278
37:sigma(2)             1.4064406       0.4035213       1.3377597       1.2411260
38:rho(1)               0.7905274       0.1768200       0.8368161       0.9482054
39:-2log Likelihood     5077.2822       7.7659076       5076.7461       5075.4617
   -2log Likelihood for means of beta estimates =   5054.9900
   DIC =   293.66354
```

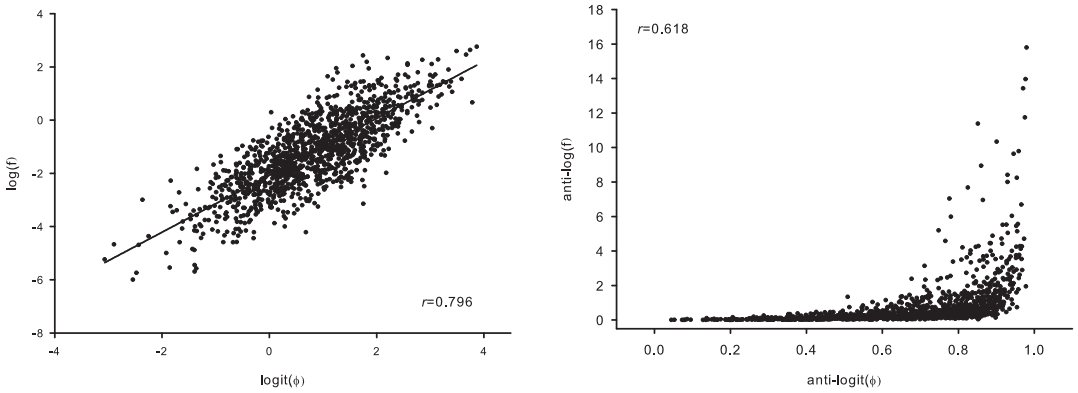
We see that the estimate of the correlation, based on the mean of the posterior, is $\hat{\rho} = 0.7905$ (the estimated median was only marginally higher). Of note, here, is that the 95% credibility interval [0.332, 0.991] doesn't bound 0, which suggests that our visual intuition concerning a positive covariance between survival and recruitment for these moth data might be correct (or at least somewhat more defensible) after all.

However, we need to be somewhat careful here. Earlier, we noted that the back-transformation of ρ from the logit scale to the real probability scale was generally an identity transformation (i.e., that the value of ρ on one scale was the same as the value on the other). However, we also noted that this was strictly true only if (i) the transformation was linear, or nearly so, (ii) monotonic, and (importantly here), (iii) was the same transformation applied to both parameters included in the multivariate hyperdistribution. Clearly, this is not will not always be the case case.

What about for the present example, where we've applied a logit transform to φ, and a log transform to f? While both transformations are monotonic, the log transform is clearly non-linear. Can we take our estimate of $\hat{\rho} = 0.7905$ and interpret it 'as is'? As you might expect, the answer is 'no'. A simple simulation will demonstrate the problem. If we take our MCMC results for μ and σ (above), and simulate 1,000 random samples from a bivariate normal logit-log distribution (φ on logit scale, f on log scale), with $\hat{\rho} = 0.7905$ (from above), our scatterplot looks something like Fig. (E.11a) – typical of bivariate normal data. The estimated correlation of our simulated data ($\hat{\rho} = 0.796$) is close to the value of 0.7905 estimated from the MCMC analysis.

However, when we back-transform the bivariate data from the logit scale and the log scale, respectively, to the real scale, the distribution is clearly no longer bivariate normal on the real scale (Fig. E.11b), to the point where even reporting the estimated bivariate correlation from the back-transformed data ($\hat{\rho} = 0.618$) verges on silly. Clearly, considerable care must be taken in interpreting the MCMC estimate of ρ, probably generally, but particularly if the two parameters are subjected to different transformations, especially if one or both transformations is highly non-linear over the range of the data.

Despite this complication in evaluating ρ for Pradel models, at this point, you might imagine building a model where λ is in the likelihood, and looking for a correlation between λ on one of the vital rates (say, apparent survival, φ), as an approach to partitioning variation in λ due to variation in a particular vital rate. We'll leave this and other interesting questions for you to pursue as an exercise.



(a) *logit(φ) versus log(f)* **(b)** *anti-logit(φ) versus anti-log(f)*

Figure E.11: *Scatterplots of random samples from a bivariate normal logit-log distribution for logit(φ) and log(f) with $\hat{\rho} = 0.7905$, and transformed and back-transformed scales. Values of μ and σ for each parameter are given in the text.*

E.3. caveats, warnings, and general recommendations

Using random effects generally, and MCMC in particular, as a basis for modeling collections of related parameters is a relatively long-standing approach in statistics and one that can, in many cases, be very effective. Use of the random effects approach in what we refer to generally as 'capture-recapture' is relatively new – in the nearly 10 years since the publication of the seminal paper by Burnham & White (2002) on the 'method of moments' approach, and despite the recent astronomic rise in the application of MCMC to this and related problems in estimation, there have been relatively few applications of these models to real data, despite what we believe are several interesting opportunities made available by these methods.

However, we also believe that both methodologies need to be better understood as to any potential pitfalls and as to its operating characteristics. The following is a summary of our experience to date with random effects models, particularly as implemented in **MARK**, and with MCMC (also as implemented in **MARK**). This material is abstracted from the equivalent section in Appendix D, supplemented with experience with such models since the time of that publication.

1. The 'method of moments' described in Appendix D, and as implemented in **MARK**, has been shown to perform well, especially when $\sigma^2 > 0.025$. This method may not do so well if $\sigma^2 \rightarrow 0$. However, we think it reasonable to believe that for a worthwhile study yielding good data, process variation, σ^2, will generally not be too small, relative to average sampling variation and it is for these conditions (of 'good data') that we need effective random effects inference methods.

   It is less clear how critical these issues are for MCMC estimation of σ^2, but a recent paper by White, Burnham & Barker (2009) suggests that MCMC may not be a complete solution. Their general conclusion was that MCMC did very well for estimating the parameter mean μ, but performance was mixed with respect to estimating process variance σ^2: for sparse data (i.e., ≤ 10 occasions), or when process variation was low, performance was poor. When there were sufficient data, and larger variance, performance was much improved.

2. Another issue to be aware of, as regards to estimation of the parameter σ^2, is the matter

of unequal, rather than equal length, time intervals. Let the time interval i have length Δ_i. Then we should parameterize the model as $S_i = (\psi_i)^{1/\Delta_i}$ where now each survival probability ψ_i is on the same unit time basis. It may then make biological sense to consider parameters that are a mean and variation for $\psi_1, \ldots, \psi_k$. But this may just as well not make sense, because the time intervals are intrinsically not comparable as they may be in very different times of the annual cycle. It becomes a subject matter judgement as to whether random effects analysis will be meaningful with unequal time intervals. For the moment, don't apply random effects models or variance components analysis – or MCMC – to situations where the intervals between sampling occasions are unequal.

3. A key design feature to focus on to meet the criterion of 'having good data' when applying random effects – or MCMC – is simply k, the number of estimable random effects parameters (time intervals, locations, etc.). The sample size for estimating σ^2 is k. Therefore, one must not have k too small; < 10 is too small. Even if we knew all the underlying S_i a sample of size $k < 10$ is too small for reliable inference about the variation of these parameters (even if we had a random sample of them, which is not required here). Inference performance has been shown to be acceptable when $k > 15$. The benefits (includes shrinkage estimates) of random effects models become greater as the number of underlying parameters, k, increases. And ability to estimate univariate and multivariate hyperparameters with MCMC also benefits significantly from longer time series.

4. A potential technical issue is the 'boundary effect', at least under what is basically a likelihood approach. As discussed in Burnham & White (2002), if one enforces the constraint $S < 1$ when the unbounded MLE $\hat{S} \geq 1$, then standard numerical methods used in **MARK** to get the observed information matrix fails. As a result, the estimated information matrix is incorrect for any terms concerning the $\hat{S}$ that is at the bound of 1 (and the inverse information matrix is likely wrong in all elements). Experience shows that, in this case, the resultant point estimate of σ^2 can be very different from what one gets when the survival parameter MLE's are allowed to be unbounded. The difference can be substantial. Using an identity link, Burnham & White found $\hat{\sigma}^2$ to be unbiased in many cases. With good data we rarely observe an unbounded MLE of S that exceeds 1. This might be explored in a Bayesian context, where it is easy (in a MCMC analysis) to allow S to have its distribution over an interval such as 0 to 2 (rather than 0 to 1). B&W considered this, and found a strong effect of the upper-bound on the point estimate (and entire posterior distribution) for σ^2, and for that particular S.

5. Another technical issue is accounting for the link function (transformation) when interpreting the estimates for various hyperparameters (μ, σ, ρ), which are generated on the appropriate transformed scale. For μ and σ, the back-transformation is relatively straightforward. The only complication is that you need to use the appropriate back-transformation for σ – either via the Delta method, or a numerical simulation. For multivariate hyperparameters (say, the correlation ρ between two structural parameters in a given model), you need to pay particular attention to whether or not both parameters are subjected to the same transformation, and whether the transformation is linear or not (at least over the range of the data). For situations where both parameters are estimated via MCMC on the logit scale, then $\hat{\rho}_{\text{logit}} = \hat{\rho}_{\text{real}}$. If, however, the transformations differ, and are non-linear for one or both parameters, interpretation is more complicated, and it may not be generally possible to come up with an acceptable interpretation of the correlation on the real scale.

E.4. Summary

This appendix has considered using Markov Chain Monte Carlo (MCMC) to estimate (i) mean and variance of a set of parameters, and (ii) covariation between sets of parameters. The former can also be accomplished using the 'method of moments' approach introduced in Appendix D, such that MCMC and 'method of moment' are best considered as complimentary approaches. In contrast, estimating the covariance among parameters requires something like MCMC.

At present, application of MCMC in **MARK** is limited to these two objectives. For more general application of MCMC to data from marked individuals, you will need to consider **BUGS**, or the equivalent. Nonetheless, the ability in **MARK** to quickly and easily explore patterns of variation and covariation among structural parameters for the large number of different data types available in **MARK** is a significant advance.

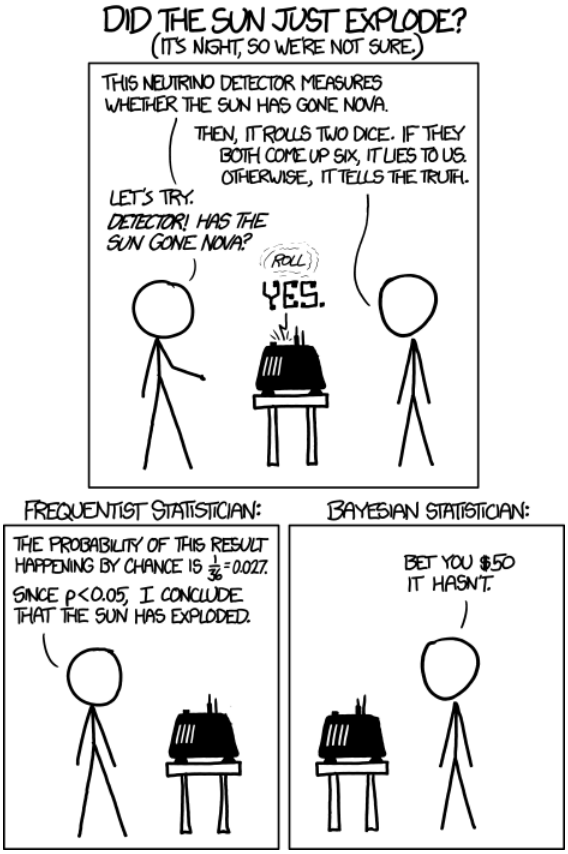
E.5. Literature

Burnham, K. P., and G. C. White. 2002. Evaluation of some random effects methodology applicable to bird ringing data. *Journal of Applied Statistics*, **29**, 245-264

White, G. C., K. P. Burnham, and R. J. Barker. 2009. Evaluation of a Bayesian MCMC random effects inference methodology for capture-mark-recapture data. Pages 1119-1127, in D. L. Thomson, E. G. Cooch, and M. J. Conroy, editors. *Modeling Demographic Processes in Marked Populations*. Springer, Berlin.

Addendum – mechanics + basic principles of MCMC

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Praesent urna sem, suscipit id aliquam id, mollis quis tellus. Phasellus varius velit a varius laoreet. Curabitur ullamcorper ante in lorem rhoncus, a sagittis orci venenatis. Quisque eu porttitor velit. Nulla auctor pharetra enim, nec feugiat eros adipiscing in. Donec lacinia neque vehicula, accumsan erat eget, interdum nibh. Aenean rutrum nec nulla nec tempor. Morbi mollis porttitor tortor. Curabitur at vestibulum ligula. Maecenas tincidunt dignissim nunc, quis cursus magna dictum et. Vestibulum sodales velit et mauris vehicula, non malesuada turpis pulvinar.



(*This addendum still under construction...stay tuned.*)

APPENDIX F

Parameter identifiability by data cloning...

There are 2 reasons why a particular model parameter might not be identifiable. The first is because the parameter may be confounded with 1 or more other parameters in the model. An example is the last φ and p parameters in a time-specific Cormack-Jolly-Seber model, where only the product of φ and p can be estimated, but not the unique values of each. In this case, the parameters are not identifiable because of the *structure* of the model. This is referred to as *intrinsic non-identifiability*. The second situation arises either because the data are inadequate, or as an artifact of the parameter being 'poorly estimated' near either the 0 or 1 boundaries. This is referred to as *extrinsic non-identifiability*. While there has been significant progress in formal 'analytical' analysis of intrinsic identifiability (see Gimenez *et al*. 2004, and Hunter & Caswell 2009, and references therein), these methods are complex, and do not apply generally to problems related to inadequate data or parameters estimated near the boundary. Cloning the data is a numerical approach which can be used generally to help identify parameters that are not estimable, for either reason (Lele *et al*. 2007, Lele *et al*. 2010).

To apply this approach, the data are *cloned* by including multiple copies of the encounter histories, i.e., duplicating the encounter histories. In **MARK**, all that needs to be done is to multiply the encounter history frequencies of each group by the number of clones desired. Consider the example of cloning the data 100 times. An encounter history for an analysis with 2 groups and no individual covariates that looks like this

```
11001010010 3 2;
```

could be cloned 100 times by entering the following encounter history:

```
11001010010 300 200;
```

By cloning the data, the sample size is increased without changing the parameter estimates. So, if the original estimates are compared to the cloned estimates, the values of the estimates will remain the same for parameters that are not confounded and are otherwise properly estimated.

How does this help us with problems of 'parameter identifiability'? Here is the key logic step – because the sample size has been increased by cloning, the standard errors of the cloned estimates will be smaller than the original standard errors. The expected result for parameters that are estimable is

$$\text{SE(original)} = \text{SE(cloned)} \times (\text{number of clones})^{0.5}$$

As an example, if the data are cloned 100 times, then the expected standard errors of the cloned data will be 1/10 of the original standard errors. The key word here is 'expected' – if in fact the estimated

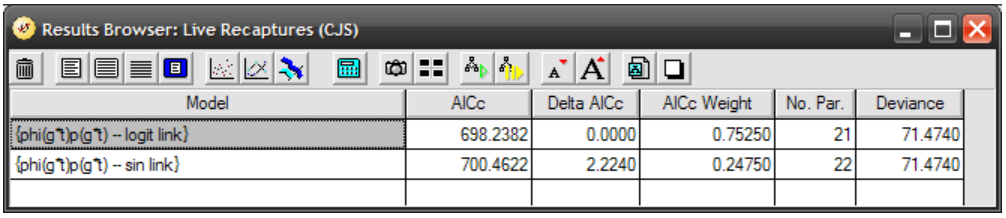
standard errors are not a fraction of the original values (by some proportion related to the size of the cloned sample), then this suggests that there may be a problem with parameter identifiability.

F.1. worked example (1) – the Dippers

The following example will use the European Dipper data (contained in `ed.inp`). Recall that the dipper data consist of 2 groups (males and females) with 7 encounter occasions. So, for a fully time-dependent model, $\{\varphi_{g*t}p_{g*t}\}$ model, we expect 22 estimable parameters: 5 survival probabilities for males, 5 survival probabilities for females, 5 encounter probabilities for males, 5 encounter probabilities for females, and 2 confounded estimates of φ_6 and p_7 for each of the 2 groups.

Let's run model $\{\varphi_{g*t}p_{g*t}\}$, first using the sin link, and then again using the logit link.

Results Browser: Live Recaptures (CJS)

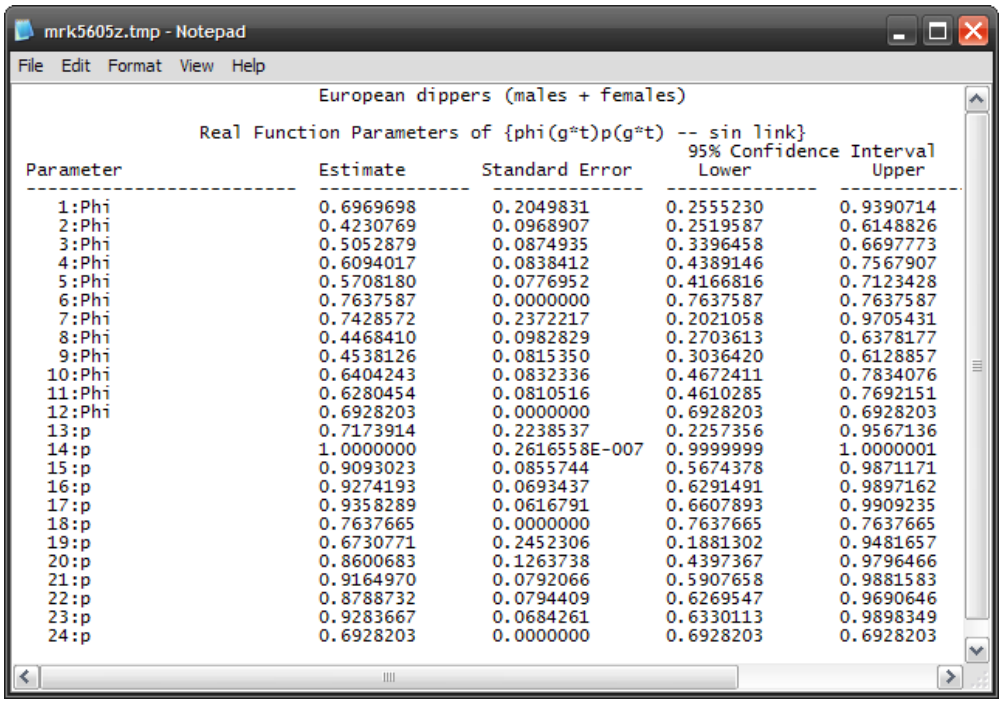


Model	AICc	Delta AICc	AICc Weight	No. Par.	Deviance
{phi(g*t)p(g*t) -- logit link}	698.2382	0.0000	0.75250	21	71.4740
{phi(g*t)p(g*t) -- sin link}	700.4622	2.2240	0.24750	22	71.4740

We see that both models have the same model deviance (71.4740), but report different number of estimated parameters. Recall from earlier chapters that the 'sin link tends to do better at estimating parameters near the boundaries than the logit link'. Thus, we might suspect that the difference in the number of reported parameters between the two link functions is due to at least one parameter being estimated near the boundary. Let's look at the estimates themselves. From the model fit using the sin link,

mrk5605z.tmp - Notepad

File Edit Format View Help



European dippers (males + females)

Real Function Parameters of {phi(g*t)p(g*t) -- sin link}

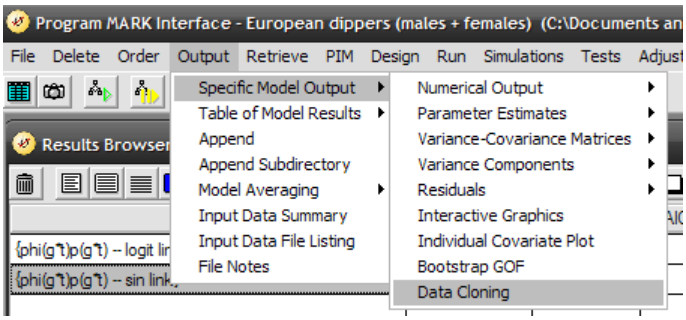
Parameter	Estimate	Standard Error	95% Confidence Interval Lower	Upper
1:Phi	0.6969698	0.2049831	0.2555230	0.9390714
2:Phi	0.4230769	0.0968907	0.2519587	0.6148826
3:Phi	0.5052879	0.0874935	0.3396458	0.6697773
4:Phi	0.6094017	0.0838412	0.4389146	0.7567907
5:Phi	0.5708180	0.0776952	0.4166816	0.7123428
6:Phi	0.7637587	0.0000000	0.7637587	0.7637587
7:Phi	0.7428572	0.2372217	0.2021058	0.9705431
8:Phi	0.4468410	0.0982829	0.2703613	0.6378177
9:Phi	0.4538126	0.0815350	0.3036420	0.6128857
10:Phi	0.6404243	0.0832336	0.4672411	0.7834076
11:Phi	0.6280454	0.0810516	0.4610285	0.7692151
12:Phi	0.6928203	0.0000000	0.6928203	0.6928203
13:p	0.7173914	0.2238537	0.2257356	0.9567136
14:p	1.0000000	0.2616558E-007	0.9999999	1.0000001
15:p	0.9093023	0.0855744	0.5674378	0.9871171
16:p	0.9274193	0.0693437	0.6291491	0.9897162
17:p	0.9358289	0.0616791	0.6607893	0.9909235
18:p	0.7637665	0.0000000	0.7637665	0.7637665
19:p	0.6730771	0.2452306	0.1881302	0.9481657
20:p	0.8600683	0.1263738	0.4397367	0.9796466
21:p	0.9164970	0.0792066	0.5907658	0.9881583
22:p	0.8788732	0.0794409	0.6269547	0.9690646
23:p	0.9283667	0.0684261	0.6330113	0.9898349
24:p	0.6928203	0.0000000	0.6928203	0.6928203

we see that parameter 6 (male survival probability, final interval) is confounded with parameter 18 (male encounter probability, final occasion), and that parameter 12 (female survival probability, final interval) is confounded with parameter 24 (female encounter probability, final occasion). In fact, if you look at the parameter estimates for the model fit using the logit link, you'll see essentially the same thing.

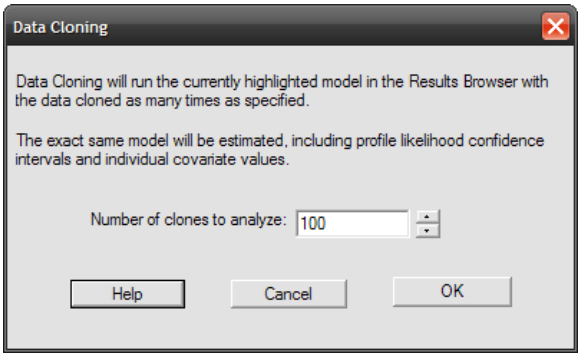
So why 22 reported parameters for the model fit with the sin link, and only 21 for the model fit with the logit link? The answer lies with parameter 14 – encounter probability for the 3rd occasion for males. We see (from the estimates shown at the bottom of the preceding page) that this parameter is estimated at 1.000, with a SE of zero (effectively). If we look at the estimates from the model fit using the logit link, we see essentially the same thing. However, because of the differences in the shape of the respective link functions near the boundaries (in this case, near the 1.0 boundary), **MARK** is unable to derive a robust estimate for survival, and thus, there is some uncertainty as to whether or not survival should be estimated at 1.0, or if this is an artifact of estimation using a particular link function. Specific details on how **MARK** numerically 'counts' parameters are given in the addendum to Chapter 4, and won't be duplicated here. However, what does remain is the question of whether or not this boundary estimate for $\hat{p}_{3,\text{male}} = 1$, or not.

F.1.1. structural identifiability – confounded parameters

One approach to resolve this problem, specifically, and the problem of parameter identifiability generally is to use data cloning. This is straightforward in **MARK**. First, highlight the model you want to use for estimation in the results browser, i.e., $\{\varphi_{g*t}p_{g*t}\}$. For now, select the model fit using the sin link. Then, select the '**`Output | Specific Model Output | Data Cloning`**' menu choice.



This will generate a pop-up menu asking you to specify the number of clones to analyze. The default of 100 is generally sufficient.



Once you click the '**OK**' button, **MARK** will proceed to fit the same model to the cloned data – the results of the model fit to the cloned data, and the estimates of the same model fit to the original data (i.e., your original analysis) will be exported to a single Excel spreadsheet:

	A	B	C	D	E	F	G	H	I	J	K
1	Index	Label	Estimate	SE	LCB	UCB	Estimate)	SE X 100	LCB X 100	UCB X 100	SE Ratio
2	1	Phi	0.69697	0.204983	0.352111	1	0.69697	0.020498	0.657559	0.738037	10.00002
3	2	Phi	0.423077	0.096891	0.246976	0.613749	0.423077	0.009689	0.40417	0.442135	10.00001
4	3	Phi	0.505288	0.087494	0.341276	0.682289	0.505288	0.008749	0.488191	0.522483	10.00001
5	4	Phi	0.609402	0.083841	0.445458	0.774226	0.609402	0.008384	0.592954	0.625815	10.00001
6	5	Phi	0.570818	0.077695	0.419989	0.722316	0.570818	0.00777	0.555586	0.586037	10
7	6	Phi	0.763759	0	0.442094	1	0.763759	15.00366	0.569346	1	0
8	7	Phi	0.742857	0.237222	0.350893	1	0.742857	0.023722	0.697331	0.790467	10.00002
9	8	Phi	0.446841	0.098283	0.274397	0.675965	0.446841	0.009828	0.427794	0.46633	10
10	9	Phi	0.453813	0.081535	0.302868	0.618985	0.453813	0.008154	0.437899	0.469854	9.999996
11	10	Phi	0.640424	0.083234	0.477012	0.804037	0.640424	0.008323	0.624093	0.656717	10
12	11	Phi	0.628045	0.081052	0.469938	0.79321	0.628045	0.008105	0.612161	0.643933	10
13	12	Phi	0.69282	0	0.345022	1	0.69282	6.915748	0.466166	1	0
14	13	p	0.717391	0.223854	0.289025	0.980072	0.717391	0.022385	0.672597	0.760115	10.00002
15	14	p	1	0	0.731975	1	1	1.74E-09	0.996718	1	0
16	15	p	0.909302	0.085574	0.666237	0.994537	0.909302	0.008557	0.891629	0.925162	9.999997
17	16	p	0.927419	0.069344	0.723089	0.995678	0.927419	0.006934	0.913062	0.940242	9.999997
18	17	p	0.935829	0.061679	0.750786	0.9962	0.935829	0.006168	0.923043	0.947222	10.00001
19	18	p	0.763767	0	0.442094	1	0.763766	15.00381	0.569346	1	0
20	19	p	0.673077	0.245231	0.246829	0.975782	0.673077	0.024523	0.624421	0.720265	10.00002
21	20	p	0.860068	0.126374	0.539856	0.991226	0.860068	0.012637	0.834186	0.88367	10.00001
22	21	p	0.916497	0.079207	0.688034	0.994995	0.916497	0.007921	0.900121	0.931163	9.999996
23	22	p	0.878873	0.079441	0.676142	0.978594	0.878873	0.007944	0.862749	0.893878	10
24	23	p	0.928367	0.068426	0.726786	0.995733	0.928367	0.006843	0.9142	0.94102	10
25	24	p	0.69282	0	0.345022	1	0.69282	6.915747	0.466166	1	0
26											

Now, step back a few pages and remember the key 'logic step' – because the sample size has been increased by cloning, the standard errors of the cloned estimates will be smaller than the original standard errors. The expected result for parameters that are estimable is

$$\text{SE(original)} = \text{SE(cloned)} \times (\text{number of clones})^{0.5}$$

In this example, the data were cloned 100 times. Thus, the expected standard errors of the cloned data should be 1/10 of the original standard errors, such that the ratio of the original SE to the cloned SE should be exactly 10. If this ratio is not close to 10 then this suggests that there may be a problem with parameter identifiability.

The ratio of the original estimated SE to the SE estimated from the cloned data is given in spreadsheet column 'K' (labeled as the '`SE Ratio`'). We see that for many – but not all – parameters, the ratio is ≈ 10. Let's focus on those ratios which are not particularly close to 10 (highlighted in the spreadsheet). For parameters 6, 12, 18, and 24 the values of the SE Ratio are ≠ 10. We recall that these are the 'confounded' parameters mentioned earlier. We refer to these as *intrinsically* non-identifiable, because the structure of the model prevents them from being identifiable.

F.1.2. 'boundary problems' – data limits and link functions

All the rest of the parameters in the spreadsheet at the top of the preceding page show a SE Ratio = 10 to at least 5 decimal places, except parameter 14, which happens to have been estimated at its boundary, i.e. 1.0. We return here to the question we noted before – is this parameter truly estimable as 1.0, or is it being estimated at 1.0 because (i) it is near the boundary, and (ii) the data, and the link function, are insufficient to resolve the parameter. To differentiate between the two (i.e., to confirm that this parameter is truly estimable), we need to compare its *profile confidence intervals* for the original and cloned data sets. A parameter at a boundary, e.g., a survival estimate equal to 1, will generally have a zero (or at least unrealistically small) standard error. Cloning the data does not change this small standard error.

However, if you have computed *profile likelihood confidence intervals* for this parameter, the profile likelihood confidence intervals for the cloned data will be considerably shorter (assuming you clone a 100 copies) than the original data. So, data cloning is also useful for verifying that a parameter estimated at the boundary is also estimable.

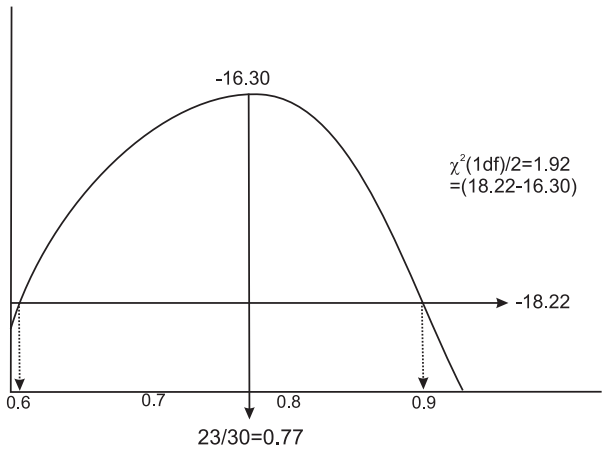
begin sidebar

profile likelihoods: a (brief) re-introduction

The classic MLE approach to variance calculation (for purposes of generating CI) is to use the negative inverse of the 2nd derivative of the MLE evaluated at the MLE. However, the problem with this approach is that it leads to derivation of symmetrical 95% CI which can yield nonsensical results, – especially for parameters that are bounded [0, 1] (e.g., UCI>1).

For example, suppose we release 30 animals, and find 1 survivor. We know from last time that the MLE for the survival probability is 1/30 = 0.03333. We also know (from Chapter 1) that the classical estimator for the variance, based on the 2nd derivative, is $\widehat{\text{var}}(\hat{p}) = \hat{p}(1-\hat{p})/N = 0.001074$. So, based on this, the 95% CI using classical approaches would be ±1.96(SE), where the SE = square-root of the variance. Thus, given var = 0.001074, the 95% CI would be ±1.96(0.03277), or [−0.031, 0.098]. Clearly nonsensical, since the LCI<0.

Fortunately, there is a better way, using something called the *profile likelihood* approach, which makes more explicit use of the shape of the likelihood. Consider the following diagram, which shows the maximum part of the log likelihood for φ, given $N = 30$, $y = 23$ (i.e., 23/30 survive).



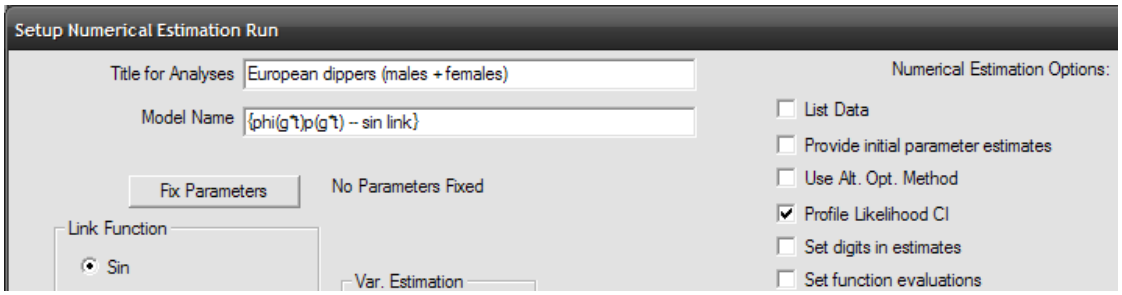
Profile likelihood confidence intervals are based on the log-likelihood function. For a single parameter, likelihood theory shows that the 2 points 1.92 units down from the maximum of the log likelihood function provide a 95% confidence interval when there is no extra-binomial variation (i.e., when $c = 1$). The value 1.92 is half of the critical value $\chi^2_1 = 3.84$. Thus, the same confidence interval

can be computed with the *deviance* by adding 3.84 to the minimum of the deviance function, where the deviance is the log-likelihood multiplied by -2 minus the -2 log likelihood value of the saturated model (for an introduction to saturated models, consult Chapter 5).

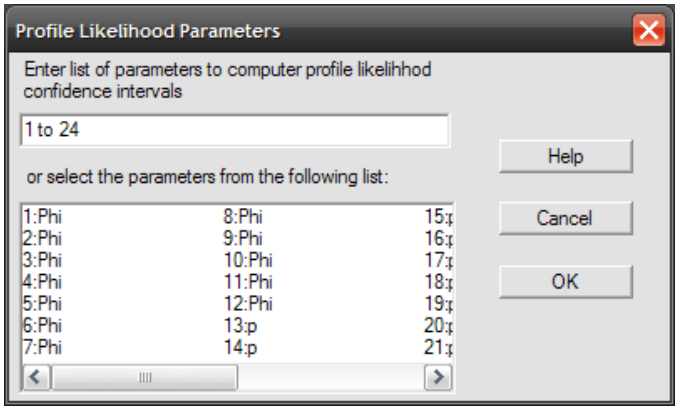
Put another way, we use the critical χ^2 value of 1.92 to derive the *profile* - you take the value of the log likelihood at the maximum (for this example, the maximum occurs at −16.30), add 1.92 to it (yielding −18.22 - note we keep the negative sign here), and look to see where the −18.22 line intersects with the *profile* of the log likelihood function. In this case, we see that the intersection occurs at approximately 0.6 and 0.9. The MLE is 23/30 = 0.767, so clearly, the profile 95% CI is not symmetrical around the MLE. But, it **is** bounded [0, 1]. The profile likelihood is the preferred approach to deriving 95% CI. The biggest limit to using it is computational - it simply takes more work (and computational time) to derive it.

end sidebar

To specify the use of profile likelihoods, you first select the model you want to clone in the results browser, $\{\varphi_{g*t}p_{g*t}\}$. Then, re-run the model, this time checking the box specifying '**`profile likelihood CI`**' on the right-hand side of the numerical estimation specification window:



You might add the phrase '`profile CI`' to the title, so you can identify the model when it is added to the browser. When you click the '**`OK`**' button, you'll be asked to specify which parameters you want to estimate the profile likelihood CI for. For this example, we'll specify all the structural parameters in the model (1 to 24).



We can see from the results browser (below) that the model fit (i.e., deviance) is identical (in other words, changing the way the CI are estimated doesn't change anything else about the model).

Results Browser: Live Recaptures (CJS)

Model	AICc	Delta AICc	AICc Weight	No. Par.	Deviance
{phi(g*t)p(g*t) -- logit link}	698.2382	0.0000	0.60321	21	71.4740
{phi(g*t)p(g*t) -- sin link}	700.4622	2.2240	0.19840	22	71.4740
{phi(g*t)p(g*t) -- sin link -- profile CI}	700.4622	2.2240	0.19840	22	71.4740

A quick comparison of the CI's estimated (for the first 4 survival parameters) using the standard 95% approach

mrk7279z.tmp - Notepad

File Edit Format View Help

European dippers (males + females)

Real Function Parameters of {phi(g*t)p(g*t) -- sin link}

Parameter	Estimate	Standard Error	95% Confidence Interval Lower	Upper
1:Phi	0.6969698	0.2049831	0.2555230	0.9390714
2:Phi	0.4230769	0.0968907	0.2519587	0.6148826
3:Phi	0.5052879	0.0874935	0.3396458	0.6697773
4:Phi	0.6094017	0.0838412	0.4389146	0.7567907

and the profile likelihood CI method

mrk8562z.tmp - Notepad

File Edit Format View Help

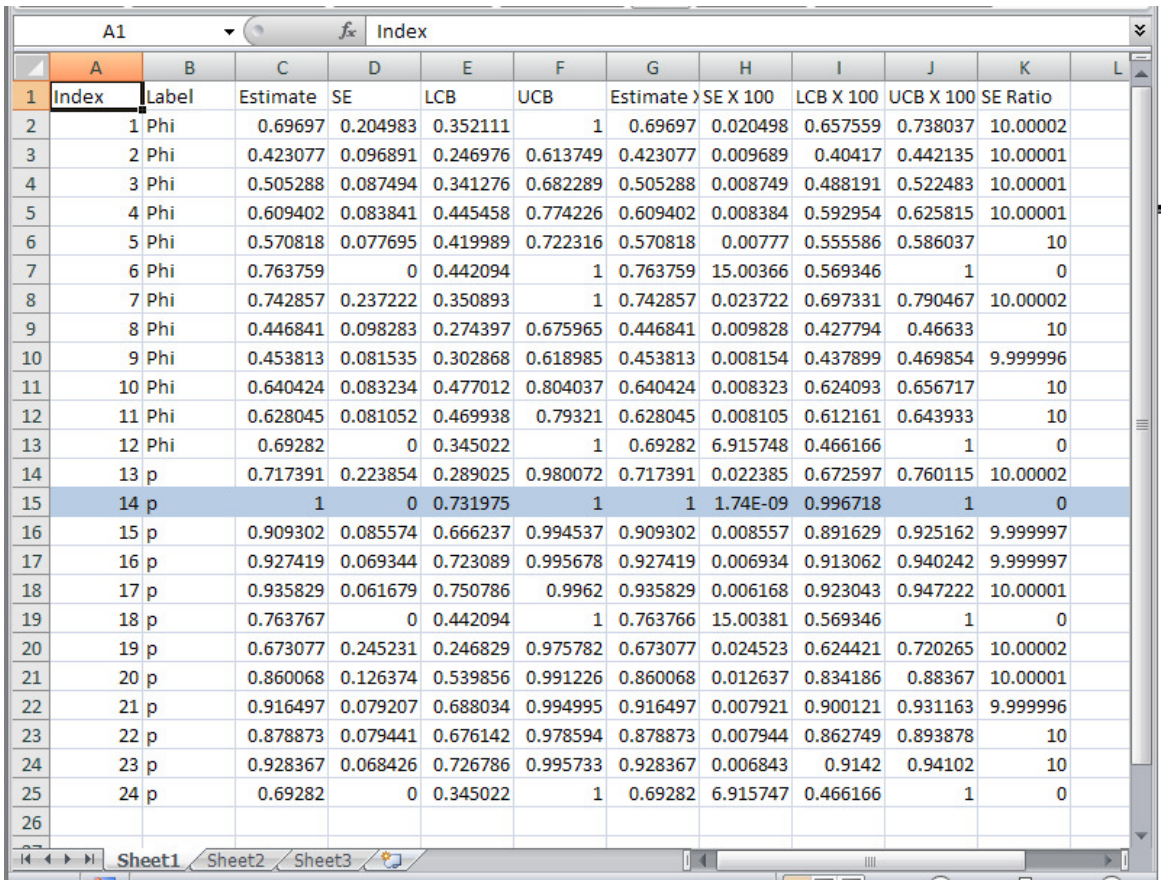
European dippers (males + females)

Real Function Parameters of {phi(g*t)p(g*t) -- sin link -- profile CI}

Parameter	Estimate	Standard Error	95% Confidence Interval Lower	Upper	
1:Phi	0.6969698	0.2049831	0.3521109	1.0000000	Profile c-hat=1.0000
2:Phi	0.4230769	0.0968907	0.2469757	0.6137493	Profile c-hat=1.0000
3:Phi	0.5052879	0.0874935	0.3412763	0.6822890	Profile c-hat=1.0000
4:Phi	0.6094017	0.0838412	0.4454577	0.7742259	Profile c-hat=1.0000

shows clear differences between the two methods for several of the parameters.

OK, back to our problem – parameter estimability. Now that we have the model fit using the profile likelihood CI in the browser, retrieve it to make sure that it is active. Then, select '**`Output | Specific Model Output | Data Cloning`**', and use the default of 100 clones. This will take a bit longer than the first time you 'analyzed the cloned data' (since profile likelihood CI take significantly longer to estimate than the classical CI – which is why it is not the default procedure in **MARK**). As before, once completed, **MARK** will export the results from both the original analysis and the analysis of the cloned data into an Excel spreadsheet (shown at the top of the next page). We are particularly interested in the CI for parameter 14 (highlighted in the spreadsheet), which we identified earlier as being 'problematic' – is the encounter probability really 1.0, or is that an artifact of the data and the link function used?



Index	Label	Estimate	SE	LCB	UCB	Estimate	SE X 100	LCB X 100	UCB X 100	SE Ratio
1	Phi	0.69697	0.204983	0.352111	1	0.69697	0.020498	0.657559	0.738037	10.00002
2	Phi	0.423077	0.096891	0.246976	0.613749	0.423077	0.009689	0.40417	0.442135	10.00001
3	Phi	0.505288	0.087494	0.341276	0.682289	0.505288	0.008749	0.488191	0.522483	10.00001
4	Phi	0.609402	0.083841	0.445458	0.774226	0.609402	0.008384	0.592954	0.625815	10.00001
5	Phi	0.570818	0.077695	0.419989	0.722316	0.570818	0.00777	0.555586	0.586037	10
6	Phi	0.763759	0	0.442094	1	0.763759	15.00366	0.569346	1	0
7	Phi	0.742857	0.237222	0.350893	1	0.742857	0.023722	0.697331	0.790467	10.00002
8	Phi	0.446841	0.098283	0.274397	0.675965	0.446841	0.009828	0.427794	0.46633	10
9	Phi	0.453813	0.081535	0.302868	0.618985	0.453813	0.008154	0.437899	0.469854	9.999996
10	Phi	0.640424	0.083234	0.477012	0.804037	0.640424	0.008323	0.624093	0.656717	10
11	Phi	0.628045	0.081052	0.469938	0.79321	0.628045	0.008105	0.612161	0.643933	10
12	Phi	0.69282	0	0.345022	1	0.69282	6.915748	0.466166	1	0
13	p	0.717391	0.223854	0.289025	0.980072	0.717391	0.022385	0.672597	0.760115	10.00002
14	p	1	0	0.731975	1	1	1.74E-09	0.996718	1	0
15	p	0.909302	0.085574	0.666237	0.994537	0.909302	0.008557	0.891629	0.925162	9.999997
16	p	0.927419	0.069344	0.723089	0.995678	0.927419	0.006934	0.913062	0.940242	9.999997
17	p	0.935829	0.061679	0.750786	0.9962	0.935829	0.006168	0.923043	0.947222	10.00001
18	p	0.763767	0	0.442094	1	0.763766	15.00381	0.569346	1	0
19	p	0.673077	0.245231	0.246829	0.975782	0.673077	0.024523	0.624421	0.720265	10.00002
20	p	0.860068	0.126374	0.539856	0.991226	0.860068	0.012637	0.834186	0.88367	10.00001
21	p	0.916497	0.079207	0.688034	0.994995	0.916497	0.007921	0.900121	0.931163	9.999996
22	p	0.878873	0.079441	0.676142	0.978594	0.878873	0.007944	0.862749	0.893878	10
23	p	0.928367	0.068426	0.726786	0.995733	0.928367	0.006843	0.9142	0.94102	10
24	p	0.69282	0	0.345022	1	0.69282	6.915747	0.466166	1	0

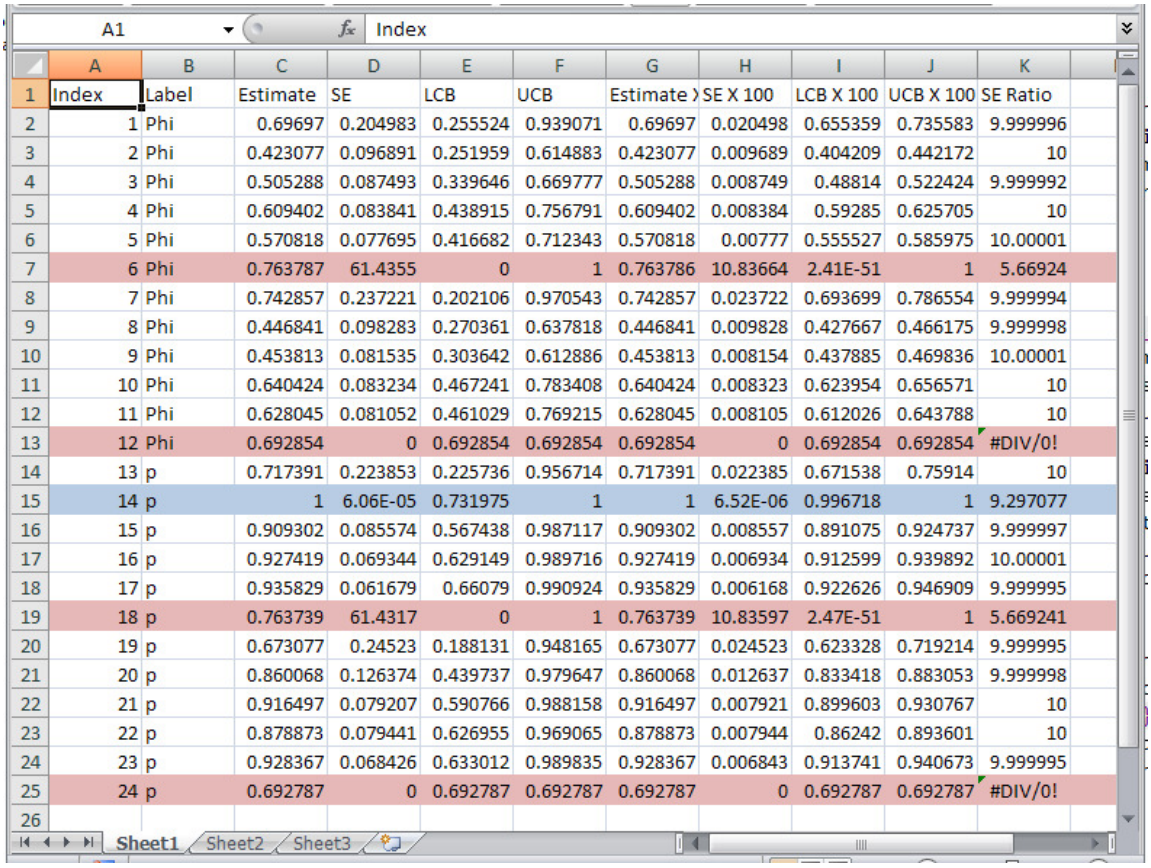
You can now see that the profile interval for parameter 14 has shortened considerably for the cloned data, with the lower bound changing from 0.732 to 0.997, indicating that this parameter was actually being estimated. In other words, parameter 14 was *extrinsically* non-identifiable. In contrast, the 4 intrinsically confounded parameters (6, 12, 18 and 24) still show a relatively wide profile likelihood confidence interval for the cloned analysis, only slightly reduced from the original values.

F.1.3. choice of link function – does it matter?

In the preceding, we considered data cloning based on model $\{\varphi_{g*t}p_{g*t}\}$, fit using the sin link. Recall that the number of reported parameters for this model differed depending on whether or not the sin (22 parameters) or the logit link was used (21 parameters). As noted earlier, the difference (for the dipper example) is due to parameter 14 – the encounter probability for the 3rd occasion for males. While neither the sin or logit link function estimate this parameter particularly 'well', because of the differences in the shape of the respective link functions near the boundaries (in this case, near the 1.0 boundary), **MARK** is both unable to derive a robust estimate of the parameter, and (more to the point), whether or not the parameter is estimable, and should be counted. Again, specific details on how **MARK** numerically 'counts' parameters are given in the addendum to Chapter 4.

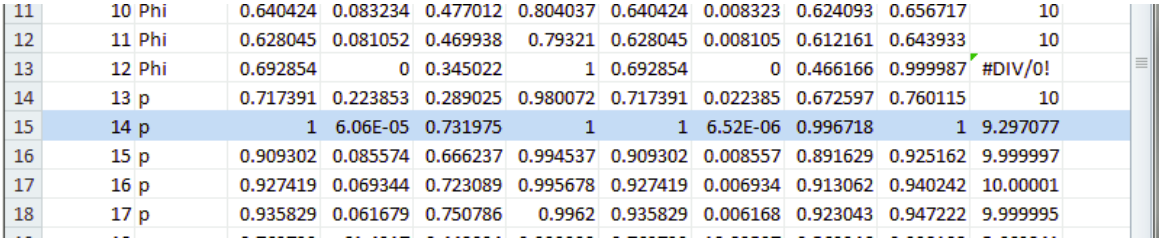
However, while this is 'interesting', our immediate interest is whether or not the choice of the link function influences the data cloning applications we're introducing here. Here, we replicate the 'cloning', but using model $\{\varphi_{g*t}p_{g*t}\}$ fit using the logit link. To do this, select the appropriate model in the

browser, and retrieve it (to make sure it is the currently 'active' model). Then, go ahead and run the data cloning as before. Here is the Excel spreadsheet containing the results. We have highlighted the confounded parameters (in red), and the 'problem' parameter 14 (in blue).



Index	Label	Estimate	SE	LCB	UCB	Estimate	SE X 100	LCB X 100	UCB X 100	SE Ratio
1	Phi	0.69697	0.204983	0.255524	0.939071	0.69697	0.020498	0.655359	0.735583	9.999996
2	Phi	0.423077	0.096891	0.251959	0.614883	0.423077	0.009689	0.404209	0.442172	10
3	Phi	0.505288	0.087493	0.339646	0.669777	0.505288	0.008749	0.48814	0.522424	9.999992
4	Phi	0.609402	0.083841	0.438915	0.756791	0.609402	0.008384	0.59285	0.625705	10
5	Phi	0.570818	0.077695	0.416682	0.712343	0.570818	0.00777	0.555527	0.585975	10.00001
6	Phi	0.763787	61.4355	0	1	0.763786	10.83664	2.41E-51	1	5.66924
7	Phi	0.742857	0.237221	0.202106	0.970543	0.742857	0.023722	0.693699	0.786554	9.999994
8	Phi	0.446841	0.098283	0.270361	0.637818	0.446841	0.009828	0.427667	0.466175	9.999998
9	Phi	0.453813	0.081535	0.303642	0.612886	0.453813	0.008154	0.437885	0.469836	10.00001
10	Phi	0.640424	0.083234	0.467241	0.783408	0.640424	0.008323	0.623954	0.656571	10
11	Phi	0.628045	0.081052	0.461029	0.769215	0.628045	0.008105	0.612026	0.643788	10
12	Phi	0.692854	0	0.692854	0.692854	0.692854	0	0.692854	0.692854	#DIV/0!
13	p	0.717391	0.223853	0.225736	0.956714	0.717391	0.022385	0.671538	0.75914	10
14	p	1	6.06E-05	0.731975	1	1	6.52E-06	0.996718	1	9.297077
15	p	0.909302	0.085574	0.567438	0.987117	0.909302	0.008557	0.891075	0.924737	9.999997
16	p	0.927419	0.069344	0.629149	0.989716	0.927419	0.006934	0.912599	0.939892	10.00001
17	p	0.935829	0.061679	0.66079	0.990924	0.935829	0.006168	0.922626	0.946909	9.999995
18	p	0.763739	61.4317	0	1	0.763739	10.83597	2.47E-51	1	5.669241
19	p	0.673077	0.24523	0.188131	0.948165	0.673077	0.024523	0.623328	0.719214	9.999995
20	p	0.860068	0.126374	0.439737	0.979647	0.860068	0.012637	0.833418	0.883053	9.999998
21	p	0.916497	0.079207	0.590766	0.988158	0.916497	0.007921	0.899603	0.930767	10
22	p	0.878873	0.079441	0.626955	0.969065	0.878873	0.007944	0.86242	0.893601	10
23	p	0.928367	0.068426	0.633012	0.989835	0.928367	0.006843	0.913741	0.940673	9.999995
24	p	0.692787	0	0.692787	0.692787	0.692787	0	0.692787	0.692787	#DIV/0!

As we saw when we applied data cloning to the model fit with the sin link, the ratio of the SE for the confounded parameters is not equal to 10, which we interpret as diagnostic of a confounding problem. For parameter 14, the ratio is not directly informative – we need to rerun the model using the profile likelihood approach.

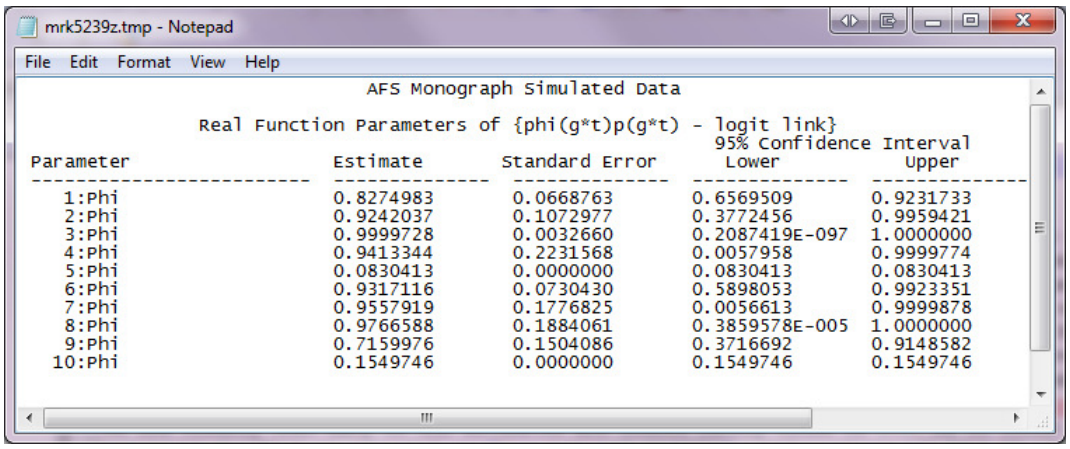


10	Phi	0.640424	0.083234	0.477012	0.804037	0.640424	0.008323	0.624093	0.656717	10
11	Phi	0.628045	0.081052	0.469938	0.79321	0.628045	0.008105	0.612161	0.643933	10
12	Phi	0.692854	0	0.345022	1	0.692854	0	0.466166	0.999987	#DIV/0!
13	p	0.717391	0.223853	0.289025	0.980072	0.717391	0.022385	0.672597	0.760115	10
14	p	1	6.06E-05	0.731975	1	1	6.52E-06	0.996718	1	9.297077
15	p	0.909302	0.085574	0.666237	0.994537	0.909302	0.008557	0.891629	0.925162	9.999997
16	p	0.927419	0.069344	0.723089	0.995678	0.927419	0.006934	0.913062	0.940242	10.00001
17	p	0.935829	0.061679	0.750786	0.9962	0.935829	0.006168	0.923043	0.947222	9.999995

As was the case when we applied this approach to the model fit with the sin link, we see that the CI for parameter 14 gets significantly smaller when the data are cloned (lower CI changes from 0.732 to 0.998). This is consistent with our determination that in fact parameter 14 is being correctly estimated at 1.000. So, in this case (and probably generally), the choice of the link function (sin or logit, typically) shouldn't make any difference.

F.2. worked example (2) – AFS monograph example

For this example we use a data set which is distributed with **MARK** (`\examples\AFSMONGR.DBF`). These live encounter data were collected over 6 sampling occasions, with 2 groups (control and treatment). The `.DBF` file distributed with **MARK** shows a series of 'standard' CJS models, fit with different link functions and design matrix structures. As in the first example, we consider model $\{\varphi_{g*t}p_{g*t}\}$. We know that there will be confounding of the final estimates of survival and encounter probability for each of the two groups. Here, we focus on the problem of estimates at or near the boundary. For these data, survival probability over an interval is anticipated to be relatively high. It is just this sort of situation which can lend itself to the 'boundary estimate' problem. If you look at the survival estimates for model $\{\varphi_{g*t}p_{g*t}\}$, fit using the logit link (shown below), we see clearly that many of the estimates are indeed fairly close to 1.0. We'll focus in particular on survival parameter 3, which is reported as $\hat{\varphi} = 0.9999729$, with a classical 95% CI of [0, 1]. Needless to say, we have significant uncertainty about the estimation of this parameter.



First, let's see if re-running the model, but using a profile likelihood CI, helps us at all. When a profile likelihood interval is requested for parameter 3, the following results are obtained: $\hat{\varphi} = 0.999728$ (essentially identical to what was reported, above). The profile CI is [0.7728344, 0.9999728]. Here we see that the optimization procedure used in generating the profile CI was able to move the LCI away from the boundary at 0. Similar results are obtained if the sin link is used.

However, our purpose here is to consider the application of data cloning to the sort of 'boundary estimate' problem. First, we re-run model $\{\varphi_{g*t}p_{g*t}\}$, using the logit link, first requesting the profile likelihood CI for parameter 3. Now, we re-run the model, with the data cloned 100 times. From the generated Excel spreadsheet, the LCI for the cloned data is reported as 0.990629. You can now see that the profile interval for parameter 3 has shortened considerably for the cloned data, with the lower bound changing from 0.773 to 0.991, indicating that this parameter was actually being estimated.

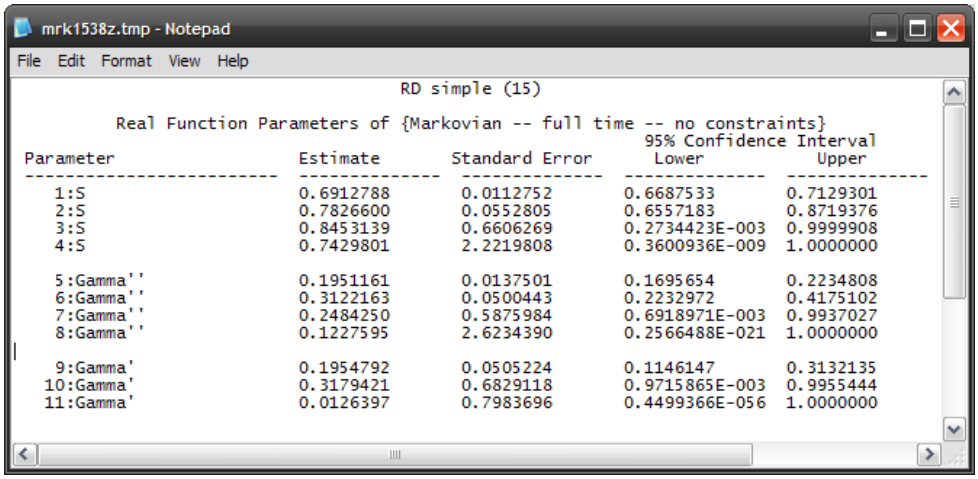
F.3. worked example (3) – robust design example

In section 15.3 of Chapter 15, we introduced an extension of the classical 'robust design' to account for temporary movements in and out of the sampled population (Kendall *et al*. 1995a, 1997). In the extended robust design, we introduced two different parameters to describe temporary movements into and out of the sample: γ' and γ'' (read as 'gamma-prime' and 'gamma-double-prime', respectively).

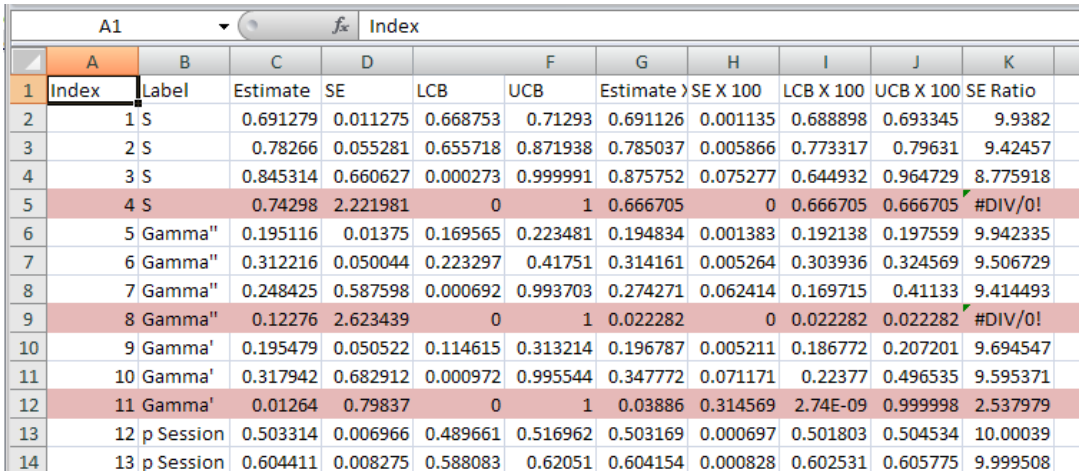
Here we are not concerned with the specific details of this model. Instead, our focus is on parameter identifiability, using the 'Markovian movement' model as an example. As discussed in Chapter 15, to provide identifiability of the parameters for the *Markovian emigration* model when parameters are time-specific, Kendall *et al*. (1997) stated that γ''_k and γ'_k need to be set equal to γ''_t and γ'_t, respectively, for some earlier period. Otherwise these parameters are confounded with S_{t-1}. They suggested setting them equal to γ''_{k-1} and γ'_{k-1}, respectively.

Can we 'demonstrate' the confounding of S_{t-1} with the γ parameters in the absence of the suggested constraints (and, by extension, can we show that the constraints do in fact 'solve' the confounding), using data cloning? To address this, we'll re-visit the 'simple robust design' example introduced in section 15.6.1 in Chapter 15 (the data are contained in `rd_simple1.inp`). From the analysis of these data described in Chapter 15, here are the survival and γ estimates from fitting a Markovian time-dependent model *without* the 'identifiability constraints' suggested by Kendall *et al*. (1997). The model was fit using a logit link.

Based on evaluation of the SE and CI of the parameter estimates (shown below), we see some 'suggestion' that some of the parameters are 'not well estimated', and may be confounded.



We can use data cloning to explore this directly. In the spreadsheet shown below



A1 | Index

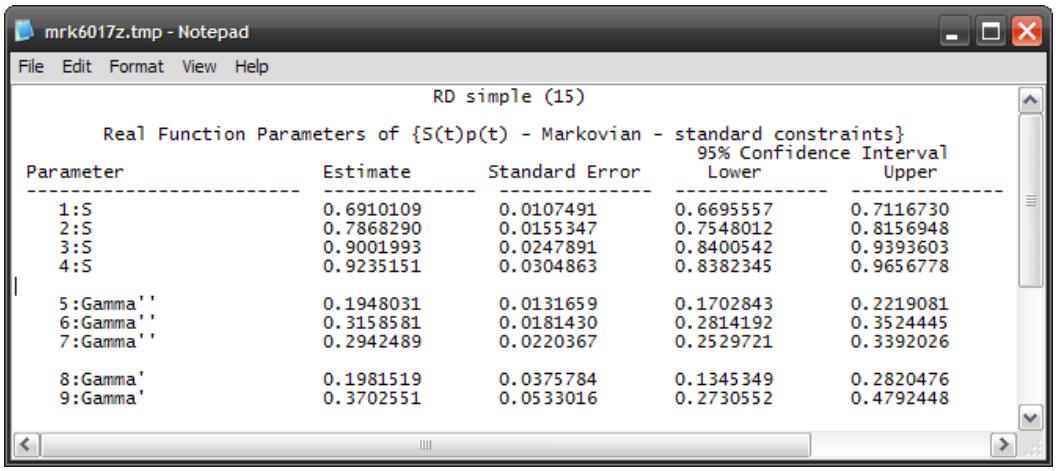
	A	B	C	D		F	G	H	I	J	K
1	Index	Label	Estimate	SE	LCB	UCB	Estimate	SE X 100	LCB X 100	UCB X 100	SE Ratio
2	1	S	0.691279	0.011275	0.668753	0.71293	0.691126	0.001135	0.688898	0.693345	9.9382
3	2	S	0.78266	0.055281	0.655718	0.871938	0.785037	0.005866	0.773317	0.79631	9.42457
4	3	S	0.845314	0.660627	0.000273	0.999991	0.875752	0.075277	0.644932	0.964729	8.775918
5	4	S	0.74298	2.221981	0	1	0.666705	0	0.666705	0.666705	#DIV/0!
6	5	Gamma''	0.195116	0.01375	0.169565	0.223481	0.194834	0.001383	0.192138	0.197559	9.942335
7	6	Gamma''	0.312216	0.050044	0.223297	0.41751	0.314161	0.005264	0.303936	0.324569	9.506729
8	7	Gamma''	0.248425	0.587598	0.000692	0.993703	0.274271	0.062414	0.169715	0.41133	9.414493
9	8	Gamma''	0.12276	2.623439	0	1	0.022282	0	0.022282	0.022282	#DIV/0!
10	9	Gamma'	0.195479	0.050522	0.114615	0.313214	0.196787	0.005211	0.186772	0.207201	9.694547
11	10	Gamma'	0.317942	0.682912	0.000972	0.995544	0.347772	0.071171	0.22377	0.496535	9.595371
12	11	Gamma'	0.01264	0.79837	0	1	0.03886	0.314569	2.74E-09	0.999998	2.537979
13	12	p Session	0.503314	0.006966	0.489661	0.516962	0.503169	0.000697	0.501803	0.504534	10.00039
14	13	p Session	0.604411	0.008275	0.588083	0.62051	0.604154	0.000828	0.602531	0.605775	9.999508

we see clear evidence that parameters S_4, γ_4'' and γ_4' (highlighted in red) are confounded. If we look closely, we also see that there is some evidence that the survival estimates might be biased, since the SE ratio for $S_1 \rightarrow S_3$ is somewhat <10, with a negative trend approaching the confounded parameter.

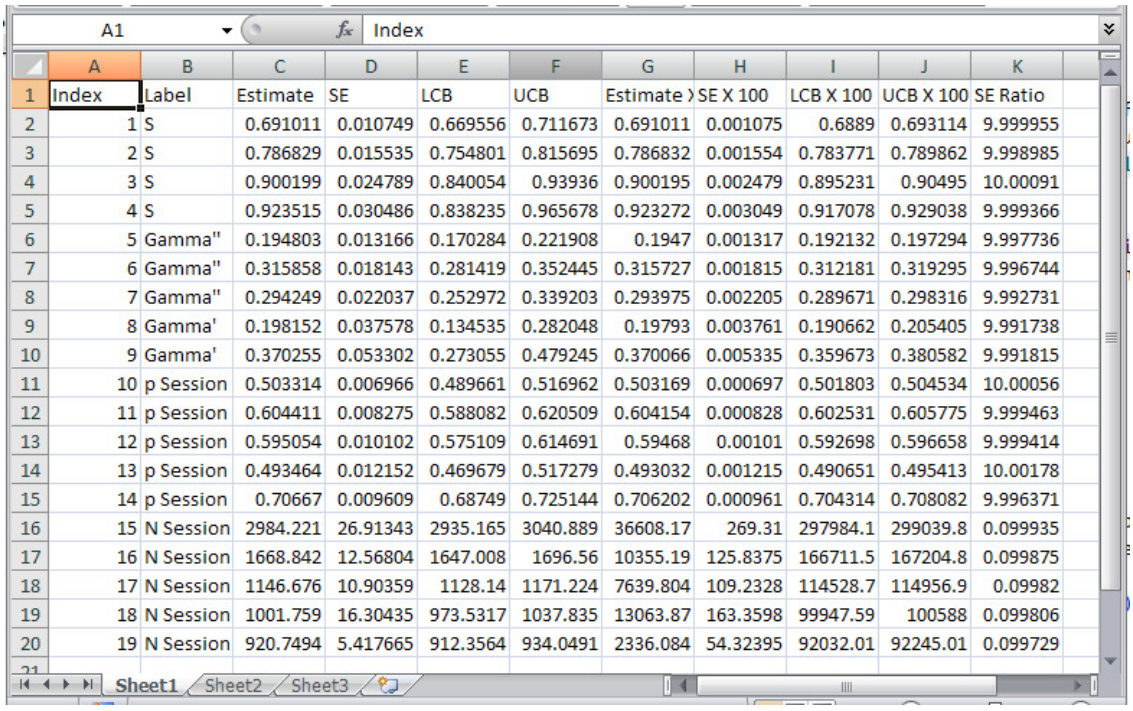
In addition, we see that the SE ratio for abundance estimates $\hat{N}$ are all $\ll 10$. Previously, we considered only parameters that are bounded $[0, 1]$. Is there an application of data cloning to estimation of a parameter that is not $[0, 1]$ bounded (say, abundance, or λ or f in a Pradel model)? First, remember that the methods described here using data cloning are to identify parameters which are confounded (i.e., intrinsically non-identifiable), or which might be poorly estimated given the data (i.e., extrinsically non-identifiable), generally because they are estimated at either the $[0, 1]$ boundary. Clearly, neither of these criterion apply to abundance N (in this example), since estimates of N are not confounded with structural parameters in any model, and since boundary issues don't apply in the usual sense (since M_{t+1} must be the lower boundary, and $M_{t+1} > 0$; see Chapter 14). Second, and perhaps more importantly, when using data cloning for $[0, 1]$ bounded parameters, only the SE changes, not the estimates themselves. This will not generally be the case for parameters where the estimates themselves are functions of the sample size – clearly, abundance is such a parameter. If you look at spreadsheet (above), you will see that the estimate of abundance using the cloned data is significantly larger than the original estimate (which of course, is not surprising, since the minimum bound of the estimate of abundance is M_{t+1}, which is multiplied by 100 during the cloning).

So, clearly, cloning should not be applied to parameters where the estimate will change as a function of cloning, and probably should not be applied to parameters that are not simple $[0, 1]$ bounded. Having said that, whether or not the cloning procedure will work for all $[0, 1]$ bounded parameters is 'a work in progress'.

Does fitting the constraint $\gamma_{k-1}'' = \gamma_k''$ and $\gamma_{k-1}' = \gamma_k'$ solve the problem of confounding with S_k? Here are the estimates for S and γ from the constrained model:



We see that, indeed, all of the parameters seem to be 'well estimated'. We can confirm this assessment by applying the data cloning approach to this model, we see (top of the next page) that all of the S and γ parameters are indeed well-estimated.

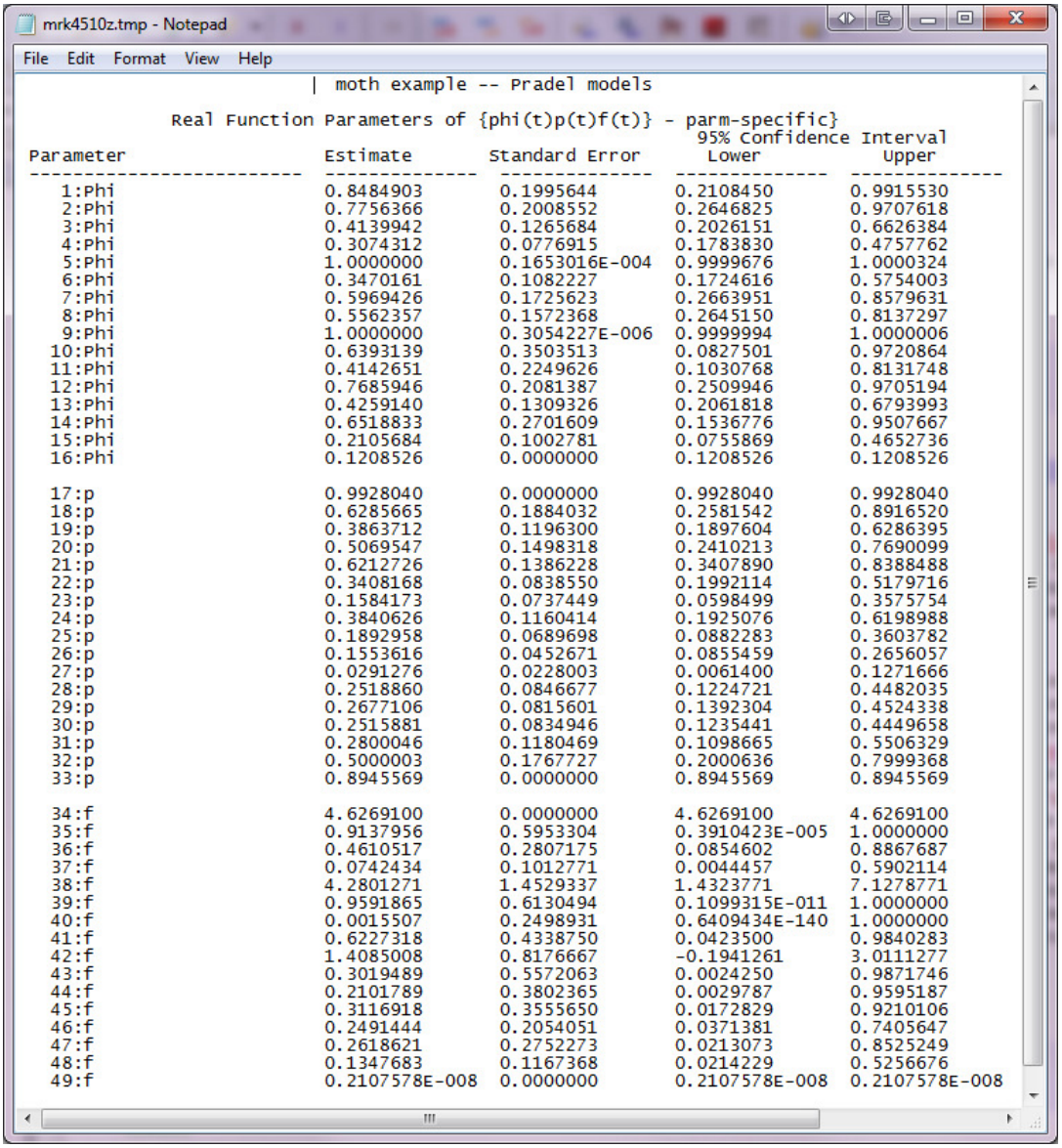


	A	B	C	D	E	F	G	H	I	J	K
1	Index	Label	Estimate	SE	LCB	UCB	Estimate	SE X 100	LCB X 100	UCB X 100	SE Ratio
2	1	S	0.691011	0.010749	0.669556	0.711673	0.691011	0.001075	0.6889	0.693114	9.999955
3	2	S	0.786829	0.015535	0.754801	0.815695	0.786832	0.001554	0.783771	0.789862	9.998985
4	3	S	0.900199	0.024789	0.840054	0.93936	0.900195	0.002479	0.895231	0.90495	10.00091
5	4	S	0.923515	0.030486	0.838235	0.965678	0.923272	0.003049	0.917078	0.929038	9.999366
6	5	Gamma"	0.194803	0.013166	0.170284	0.221908	0.1947	0.001317	0.192132	0.197294	9.997736
7	6	Gamma"	0.315858	0.018143	0.281419	0.352445	0.315727	0.001815	0.312181	0.319295	9.996744
8	7	Gamma"	0.294249	0.022037	0.252972	0.339203	0.293975	0.002205	0.289671	0.298316	9.992731
9	8	Gamma'	0.198152	0.037578	0.134535	0.282048	0.19793	0.003761	0.190662	0.205405	9.991738
10	9	Gamma'	0.370255	0.053302	0.273055	0.479245	0.370066	0.005335	0.359673	0.380582	9.991815
11	10	p Session	0.503314	0.006966	0.489661	0.516962	0.503169	0.000697	0.501803	0.504534	10.00056
12	11	p Session	0.604411	0.008275	0.588082	0.620509	0.604154	0.000828	0.602531	0.605775	9.999463
13	12	p Session	0.595054	0.010102	0.575109	0.614691	0.59468	0.00101	0.592698	0.596658	9.999414
14	13	p Session	0.493464	0.012152	0.469679	0.517279	0.493032	0.001215	0.490651	0.495413	10.00178
15	14	p Session	0.70667	0.009609	0.68749	0.725144	0.706202	0.000961	0.704314	0.708082	9.996371
16	15	N Session	2984.221	26.91343	2935.165	3040.889	36608.17	269.31	297984.1	299039.8	0.099935
17	16	N Session	1668.842	12.56804	1647.008	1696.56	10355.19	125.8375	166711.5	167204.8	0.099875
18	17	N Session	1146.676	10.90359	1128.14	1171.224	7639.804	109.2328	114528.7	114956.9	0.09982
19	18	N Session	1001.759	16.30435	973.5317	1037.835	13063.87	163.3598	99947.59	100588	0.099806
20	19	N Session	920.7494	5.417665	912.3564	934.0491	2336.084	54.32395	92032.01	92245.01	0.099729

F.4. worked example (4) – Pradel model example

Our final example re-visits the analysis of a famous set of data, the moth (*Gonodontis bidenta*) data reported on by Bishop *et al*. (1978), presented earlier in Appendix D (section D4.4). The data consist of records for 689 male moths that were captured, marked, and released daily over 17 days in northwest England. These moths were non-melanic; demographic parameters were estimated as part of a larger study looking at comparative fitness of distinct color morphs.

In Appendix D we considered the use of random effect Pradel models, focussing on estimation of process variance, and possible trend, in realized growth rate λ. Here, our interest is on using data cloning to evaluate possible intrinsic non-identifiability in time-dependent Pradel models. The data we'll work with are contained in `moth-example.inp`. We will fit a fully time-dependent model using the '**`Pradel survival & Recruitment`**' data type. Recall (from Chapter 12) that there are 3 structural parameters specifying this model: φ, p and f (where f is the per capita recruitment probability). We will go ahead and fit model $\{\varphi_t p_t f_t\}$ to the moth data, using the default PIM structure and sin link. Given 17 sampling occasions, there are 49 structural parameters in the model (16 φ, 17 p, and 16 f parameters). As an *a priori* check against possible numerical problems, we'll use the '**`Alt. Opt. Method`**' (i.e., simulated annealing). Since the recruitment parameter f is not necessarily bounded [0, 1] we'll use the log link function for the recruitment parameters. Once the numerical estimation has finished, we'll add the results to the browser. Model deviance is 196.844. Although there are 49 structural parameters in the model, **MARK** reports that only 45 are identifiable, given the structure of the model, and the data. The real parameter estimates are shown at the top of the next page.



mrk4510z.tmp - Notepad

File Edit Format View Help

moth example -- Pradel models

Real Function Parameters of {phi(t)p(t)f(t)} - parm-specific}

Parameter	Estimate	Standard Error	95% Confidence Interval Lower	Upper
1:Phi	0.8484903	0.1995644	0.2108450	0.9915530
2:Phi	0.7756366	0.2008552	0.2646825	0.9707618
3:Phi	0.4139942	0.1265684	0.2026151	0.6626384
4:Phi	0.3074312	0.0776915	0.1783830	0.4757762
5:Phi	1.0000000	0.1653016E-004	0.9999676	1.0000324
6:Phi	0.3470161	0.1082227	0.1724616	0.5754003
7:Phi	0.5969426	0.1725623	0.2663951	0.8579631
8:Phi	0.5562357	0.1572368	0.2645150	0.8137297
9:Phi	1.0000000	0.3054227E-006	0.9999994	1.0000006
10:Phi	0.6393139	0.3503513	0.0827501	0.9720864
11:Phi	0.4142651	0.2249626	0.1030768	0.8131748
12:Phi	0.7685946	0.2081387	0.2509946	0.9705194
13:Phi	0.4259140	0.1309326	0.2061818	0.6793993
14:Phi	0.6518833	0.2701609	0.1536776	0.9507667
15:Phi	0.2105684	0.1002781	0.0755869	0.4652736
16:Phi	0.1208526	0.0000000	0.1208526	0.1208526
17:p	0.9928040	0.0000000	0.9928040	0.9928040
18:p	0.6285665	0.1884032	0.2581542	0.8916520
19:p	0.3863712	0.1196300	0.1897604	0.6286395
20:p	0.5069547	0.1498318	0.2410213	0.7690099
21:p	0.6212726	0.1386228	0.3407890	0.8388488
22:p	0.3408168	0.0838550	0.1992114	0.5179716
23:p	0.1584173	0.0737449	0.0598499	0.3575754
24:p	0.3840626	0.1160414	0.1925076	0.6198988
25:p	0.1892958	0.0689698	0.0882283	0.3603782
26:p	0.1553616	0.0452671	0.0855459	0.2656057
27:p	0.0291276	0.0228003	0.0061400	0.1271666
28:p	0.2518860	0.0846677	0.1224721	0.4482035
29:p	0.2677106	0.0815601	0.1392304	0.4524338
30:p	0.2515881	0.0834946	0.1235441	0.4449658
31:p	0.2800046	0.1180469	0.1098665	0.5506329
32:p	0.5000003	0.1767727	0.2000636	0.7999368
33:p	0.8945569	0.0000000	0.8945569	0.8945569
34:f	4.6269100	0.0000000	4.6269100	4.6269100
35:f	0.9137956	0.5953304	0.3910423E-005	1.0000000
36:f	0.4610517	0.2807175	0.0854602	0.8867687
37:f	0.0742434	0.1012771	0.0044457	0.5902114
38:f	4.2801271	1.4529337	1.4323771	7.1278771
39:f	0.9591865	0.6130494	0.1099315E-011	1.0000000
40:f	0.0015507	0.2498931	0.6409434E-140	1.0000000
41:f	0.6227318	0.4338750	0.0423500	0.9840283
42:f	1.4085008	0.8176667	-0.1941261	3.0111277
43:f	0.3019489	0.5572063	0.0024250	0.9871746
44:f	0.2101789	0.3802365	0.0029787	0.9595187
45:f	0.3116918	0.3555650	0.0172829	0.9210106
46:f	0.2491444	0.2054051	0.0371381	0.7405647
47:f	0.2618621	0.2752273	0.0213073	0.8525249
48:f	0.1347683	0.1167368	0.0214229	0.5256676
49:f	0.2107578E-008	0.0000000	0.2107578E-008	0.2107578E-008

We show the full set of real parameter estimates because we want to make several points. First, we should have an intuitive sense by now that there are likely (inevitably?) going to be intrinsically non-identifiable parameters in fully time-dependent models. Typically, these show up at the 'end' of a time-series. Because the Pradel model uses encounter histories going both 'backwards' and 'forwards' through time, we might suspect there would be intrinsic non-identifiability of parameters both at the start and end of the time-series. However, here we have 3 structural parameters (φ, p and f), which may interact in unexpected ways. Finally, we probably don't anticipate that intrinsic non-identifiability will show up in the middle of a time-series for a parameter – non-identifiable parameters 'in the middle' of a time-series are more likely to reflect extrinsic non-identifiability (i.e., limitations of data, or boundary estimation problems).

Based on our estimates (above) we see clear evidence that the final estimates for φ, p and f (parameters 16, 33 and 49) are all poorly estimated (any parameter with a reported SE of 0.000 is not well estimated). However, we also see some other parameters that weren't well estimated. In particular, the

first p and f estimates (parameters 17 and 34) are also reported with SE of 0.000. Finally, several of the 'interior' estimates for φ and f (i.e., those in the middle of the time-series) are not well-estimated (parameters 5, 9, 35, 39 and 40, respectively).

We use the data cloning procedure to identify both the intrinsically and potentially extrinsically non-identifiable parameters. In the following spreadsheet, we have highlighted the intrinsically non-identifiable parameters (we've edited out some of the rows corresponding to some of the 'interior parameters' to make it fit the page).

	B	C	D	E	F	G	H	I	J	K
1	Label	Estimate	SE	LCB	UCB	Estimate)	SE X 100	LCB X 100	UCB X 100	SE Ratio
2	Phi	0.848486	0.199565	0.21085	0.991552	0.848486	0.019956	0.805106	0.883605	10.00021
3	Phi	0.775641	0.20085	0.264692	0.970762	0.775641	0.020085	0.733847	0.812549	10.00006
4	Phi	0.41399	0.126566	0.202615	0.662632	0.413993	0.012657	0.389423	0.438998	9.999946
5		<edited to save space>								
6	Phi	0.768589	0.208134	0.251007	0.970516	0.768584	0.020812	0.725317	0.806852	10.00056
7	Phi	0.425927	0.13094	0.206182	0.679423	0.425924	0.013093	0.400482	0.451765	10.00048
8	Phi	0.65188	0.270159	0.153679	0.950765	0.651878	0.027015	0.59724	0.702794	10.00017
9	Phi	0.21057	0.100281	0.075586	0.465283	0.210573	0.010028	0.19159	0.2309	9.999976
10	Phi	0.125317	0	0.125317	0.125317	0.125315	0.713065	4.16E-07	0.99998	0
11	p	0.469369	0	0.469369	0.469369	0.469369	4.603408	1.64E-16	1	0
12	p	0.628573	0.188413	0.258142	0.891663	0.628572	0.01884	0.590959	0.664689	10.00061
13	p	0.386364	0.119627	0.189758	0.628628	0.386363	0.011963	0.363196	0.410057	10.00008
14	p	0.50696	0.149838	0.241017	0.769022	0.50696	0.014984	0.477601	0.53627	10.00005
15		<edited to save space>								
16	p	0.267712	0.081559	0.139233	0.452433	0.267714	0.008156	0.252035	0.283999	10.00039
17	p	0.251582	0.083497	0.123537	0.444967	0.251583	0.008349	0.235574	0.268298	10.0004
18	p	0.280004	0.118045	0.109868	0.550627	0.280003	0.011804	0.25746	0.303712	10.00008
19	p	0.499996	0.176777	0.200056	0.79994	0.499995	0.017677	0.465403	0.534587	10.00023
20	p	0.862679	0	0.862679	0.862679	0.862677	4.90865	3.37E-35	1	0
21	f	1.740151	0	1.740151	1.740151	1.740151	25.38825	-48.0208	51.50113	0
22	f	0.913817	0.595371	3.9E-06	1	0.913831	0.059533	0.706719	0.979024	10.00063
23	f	0.461034	0.280715	0.085455	0.886761	0.461033	0.028071	0.406704	0.516303	10.00015
24		<edited to save space>								
25	f	0.074242	0.101285	0.004444	0.590274	0.074241	0.010128	0.056672	0.096699	10.00015
26	f	0.249158	0.205417	0.037139	0.740588	0.249161	0.020541	0.211103	0.291545	10.00056
27	f	0.261842	0.275226	0.021303	0.852528	0.261852	0.027522	0.211577	0.319237	10.00033
28	f	0.134771	0.116737	0.021424	0.525668	0.134767	0.011674	0.113481	0.159329	10.00014
29	f	0	0	0	0	2.09E-09	2.7E-07	-5.3E-07	5.31E-07	0

We see clearly that, as we suspected, several of the parameters are intrinsically non-identifiable. Specifically, the first and last p and f parameters, and the final φ parameter. We note that some of these parameters may become identifiable, provided the proper constraints are applied (e.g., if we fix the first and last p parameters to 1, then all of the f and φ parameters become estimable).

Next, we look for extrinsically non-identifiable parameters. We re-run our model, using profile likelihood estimation for the CI, followed by data cloning on this model run with the profile CI. Given the number of structural parameters in this model, this can take some time (and, in fact, the profile CI is not properly estimated for some parameters, due to numerical optimization issues). The results for a subset of the 'problem' parameters are shown at the top of the next page. To start, focus on parameters 5 and 9.

G23 f_x 0.340813343989751

	A	B	C	D	E	F	G	H	I	J	K
1	Index	Label	Estimate	SE	LCB	UCB	Estimate)	SE X 100	LCB X 100	UCB X 100	SE Ratio
2	1	Phi	0.848486	0.199565	0.500489	1	0.848486	0.019956	0.80998	0.888287	10.00021
3	2	Phi	0.775641	0.20085	0.473309	1	0.775641	0.020085	0.737452	0.816272	10.00006
4	3	Phi	0.41399	0.126566	0.224775	0.767104	0.413993	0.012657	0.389923	0.439582	9.999946
5	4	Phi	0.307435	0.077691	0.180211	0.502477	0.307438	0.007769	0.292495	0.322958	9.999904
6	5	Phi	1	6.93E-05	0.524986	1	1	5.99E-06	0.971486	1	11.56387
7	6	Phi	0.347019	0.108281	0.185801	0.633276	0.347019	0.010796	0.326413	0.367871	10.02943
8	7	Phi	0.596933	0.172574	0.335786	1	0.59693	0.017251	0.564157	0.631756	10.00389
9	8	Phi	0.556247	0.157239	0.313581	1	0.556247	0.015724	0.526202	0.588105	10.00016
10	9	Phi	1	0.000185	0.489078	1	1	1.75E-05	0.94974	1	10.5753
11	10	Phi	0.639277	0.350268	0.258449	1	0.639283	0.035026	0.575419	0.71342	10.00019
12	11	Phi	0.414279	0.224937	0.169009	1	0.414274	0.022492	0.371566	0.459759	10.00055

For parameters 5 and 9, we see that in both instances, the lower CI bound has 'shortened' considerably for the cloned data, indicating that these parameters were actually being estimated. In contrast, the intrinsically non-identifiable parameters still show a relatively wide profile CI for the cloned analysis, with only a slight reduction from the original values. Here, for example, and the results for the final φ and first p parameters:

15	14	Phi	0.65188	0.270159	0.301829	1	0.651878	0.027015	0.601396	0.707536	10.00017
16	15	Phi	0.21057	0.100281	0.082806	0.616568	0.210573	0.010028	0.191885	0.231303	9.999976
17	16	Phi	0.125317	0	0.034872	1	0.125315	0.713065	0.098376	1	0
18	17	p	0.469369	0	0.073434	1	0.469369	4.603408	0.144224	1	0
19	18	p	0.628573	0.188413	0.305333	0.919877	0.628572	0.01884	0.591486	0.665222	10.00061
20	19	p	0.386364	0.119627	0.203168	0.64007	0.386363	0.011963	0.363264	0.410135	10.00008

However, recall that both φ and p are bounded $[0, 1]$. How well does our 'data cloning' approach apply for parameters that are not bounded $[0, 1]$? Say, the recruitment parameter, f, in a Pradel model, where the upper limit can potentially be > 1. Look back at the parameter estimates for $\hat{f}_i$ tabulated at the top of p. F.14. We see that recruitment parameters 35, 39 and 40 seem to be extrinsically non-identifiable. The reported CI for all three estimates is effectively $[0, 1]$, which seems suspicious. The first and last estimates (parameters 34 and 49) are also intrinsically confounded (as we showed earlier). If we re-run our model using the profile likelihood CI routine, we find that there are several 'numerical problems' in estimating the profile CI for some parameters, in particular, some of the recruitment parameters. These problems were not 'solved' by using a different link function (say, the log link for the f parameters). So, we proceed cautiously, and apply the data cloning approach to the profile CI analysis. We see (below) that the profile CI reported for the cloned samples are narrower, suggesting that these parameters are being estimated correctly.

Index	Label	Estimate	SE	LCB	UCB	Estimate)	SE X 100	LCB X 100	UCB X 100	SE Ratio
34	f	4.627742	49.837	2E-07	13.00067	4.627628	10.42264	1.51E-05	5.077974	4.781611
35	f	0.913797	0.595365	0.048999	2.699406	0.913804	0.059535	0.800775	1.034633	10.00022
36	f	0.461027	0.280716	0.083079	1.340597	0.461033	0.028071	0.408324	0.51857	10.00023
37	f	0.074245	0.101287	2E-07	0.358755	0.074244	0.010128	0.054948	0.094689	10.00026
38	f	4.280204	1.452989	1.697582	8.817474	4.280152	0.145296	4.003777	4.575701	10.00022
39	f	0.959156	0.63183	0.230015	2.474151	0.959163	0.06113	0.845306	1.039754	10.33579
40	f	0.001561	0.262515	1E-07	0.81718	0.001561	0.024876	0.000446	0.001561	10.55294
41	f	0.622721	0.434489	0.071249	2.22489	0.622733	0.043378	0.541724	0.711583	10.01629

F.5. Limitations & other thoughts and approaches

As mentioned in our discussion of the 'robust design' example (section F.3), data cloning (as currently implemented in **MARK**) should not be applied to parameters where the estimate will change as a function of cloning (e.g., abundance), and probably should not be applied (or applied with some care) to parameters that are not simple $[0, 1]$ bounded (such as the recruitment parameter f in the Pradel model in the preceding example). Having said that, whether or not the cloning procedure will work for all $[0, 1]$ bounded parameters is 'a work in progress'. Stay tuned.

One limitation in the current implementation of data cloning is that the results are only reported in the Excel spreadsheet for the real parameters, and not the β parameters. However, you should be able to work backwards from the real parameters to determine which of the beta parameters are causing the confounding.

What about other 'numerical' approaches to the problem of confounding, and estimability? The two most commonly suggested are 'random effects' (Appendix D), and 'MCMC' (Appendix E). The basic idea motivating consideration of a 'random effects' approach is that since a RE model 'shrinks' the estimate toward the model-specified mean, then if a particular 'boundary estimate' is not shrunk much away from the boundary that this estimate is in fact 'correctly' estimated near the boundary. Alas, this is not correct, for both conceptual and practical reasons (see Appendix D). Specifically, the amount of shrinkage is a function of the relative magnitude of process and sampling variance for a particular parameter, which has nothing specifically to do with whether or not a parameter is estimated near a boundary.

There has also been some suggestion that MCMC (see Appendix E) is 'robust' to estimation along the boundary. Again, this is not correct, especially for monotonic link functions (like the logit). The MCMC sampler in **MARK** samples on the transformed (beta) scale, and there is no particular reason to assume it will do better at estimating the parameter (although there is an argument that it will perform well at generating a credible CI for the estimate – whether or not such an interval is 'better' than a profile likelihood CI is an open question).

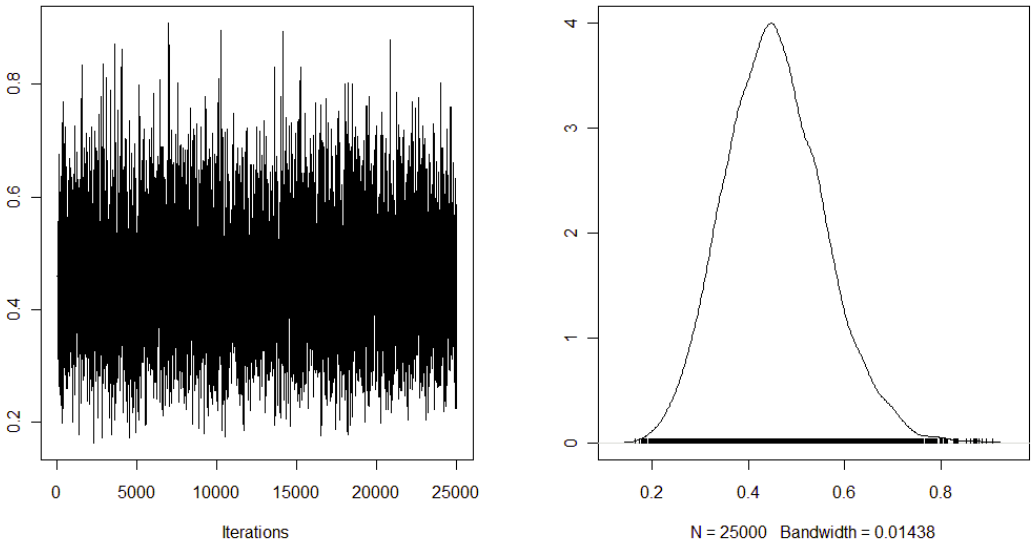
begin sidebar

using MCMC to diagnose non-identifiable parameters

While MCMC will not 'solve' the problems of non-identifiable parameters, it is possible in some cases to use MCMC as a tool to diagnose (identify) parameters which are potentially non-identifiable (Gimenez *et al*. 2009). Recall that in a Bayesian analysis, the posterior is a function of the likelihood and the prior. Now, consider intrinsic non-identifiability. In this case, the likelihood surface for the parameter is fairly 'flat' (non-informative', and the posterior will strongly reflect the prior. In the cases of extrinsic non-identifiability, there is so little 'information' in the data for a given parameter, that the posterior again will strongly reflect the prior. In other words, if the 'shape' of the posterior is not appreciably different than the prior, then there is reason to expect that the parameter is not identifiable.

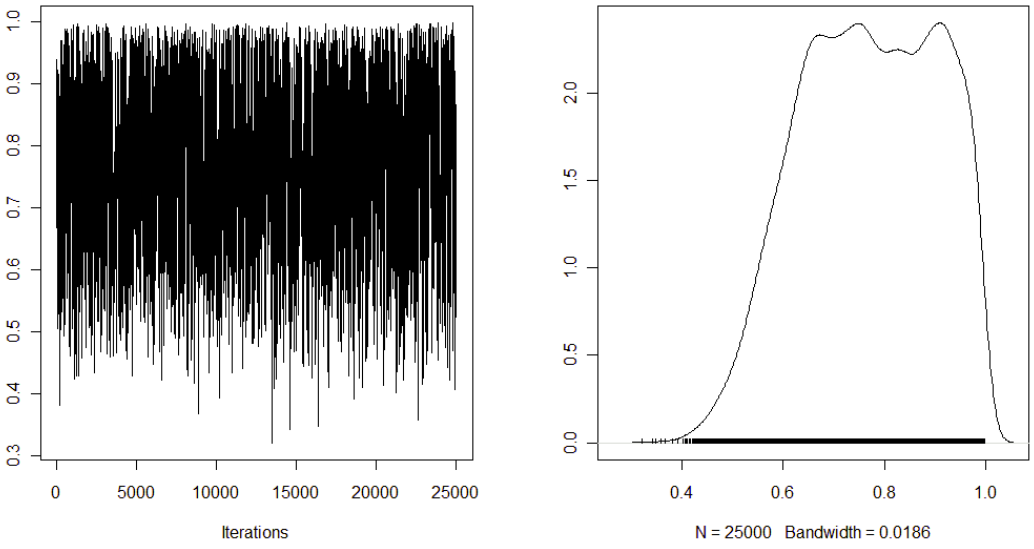
To demonstrate the basic idea, let's consider an MCMC analysis of the European Dipper data set (males and females). As discussed in section F.1, for a fully time-dependent model $\{\varphi_{s*t}p_{s*t}\}$, the final survival and encounter probability parameters φ_{k-1} and p_k are confounded for both sexes, respectively. These intrinsically non-identifiable parameters were clearly identified by the data cloning procedure. Can we achieve the same result using an MCMC approach? Skipping the details of the mechanics (see Appendix E), let's consider the trace and density plots for a couple of parameters from this model. Recall that the default prior used by **MARK** for MCMC estimation is a uniform (flat) prior.

First, consider the MCMC output for real parameter 2 (corresponding to $\varphi_{2,m}$).



We see that the trace is dense, regular, and symmetrical around the mode. The corresponding density plot is clearly unimodal, peaked, and very different in shape from the uniform ('flat') prior. This parameter is clearly identifiable, and well-estimated.

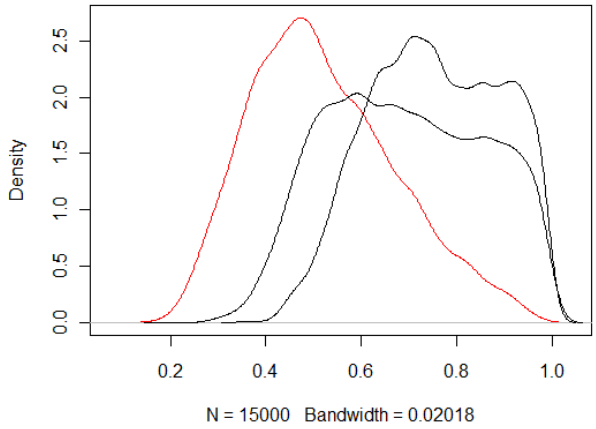
Contrast this with the trace and density plot for real parameter 6 (corresponding to $\varphi_{6,m}$), which we know to be intrinsically confounded with real parameter 12 ($p_{7,m}$).



Here, we see that not only is the trace plot rather 'strange looking' (by various subjective criteria), but (more informatively), the density plot appears 'flat' over most of the range, clearly reflecting the strong influence of the default uniform (flat) prior. The is 'diagnostic' of an intrinsically non-identifiable parameter.* Examination of the other intrinsically non-identifiable parameters for this model (real parameters 12, 18 and 24) show the same 'flat' density plots.

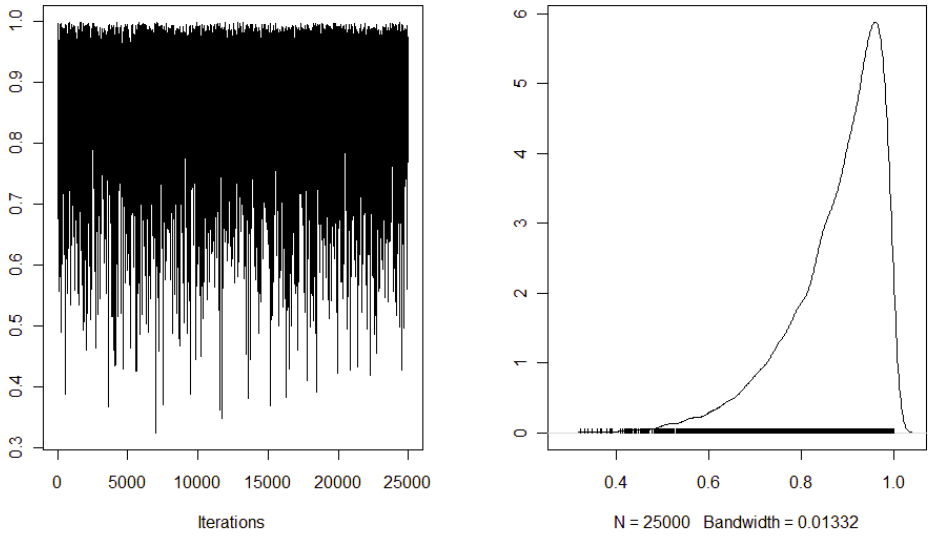
* Gimenez *et al*. (2009) present a more 'quantitative' approach to comparing the posterior density and the prior.

The underlying reason that the posterior plots for parameters that are flat for some distance across the top is that there is only a limited range for each of the parameters over which the product is defined to give exactly the same value equal to the MLE of the product (recall that **MARK** estimates a function of the product of φ and p for the confounded parameters). Further, if you plot the posterior for the product (by multiplying the 2 values for each sample from the posterior), we would get a 'pointed' posterior (the mode of which is what **MARK** reports). For example, compare the density plot (below) for the product of the posterior sample for parameters 6 ($\varphi_{6,m}$) and 12 ($p_{7,m}$), indicated by the red line, with the density plots for each parameter separately, shown by the black lines.



It is also worth noting that the width of the 'flat top' is a function of what the product value is. So, as the value of the product goes toward one, the width of the posterior for each of the pieces becomes progressively narrower, and hence your ability to detect non-identifiability declines.

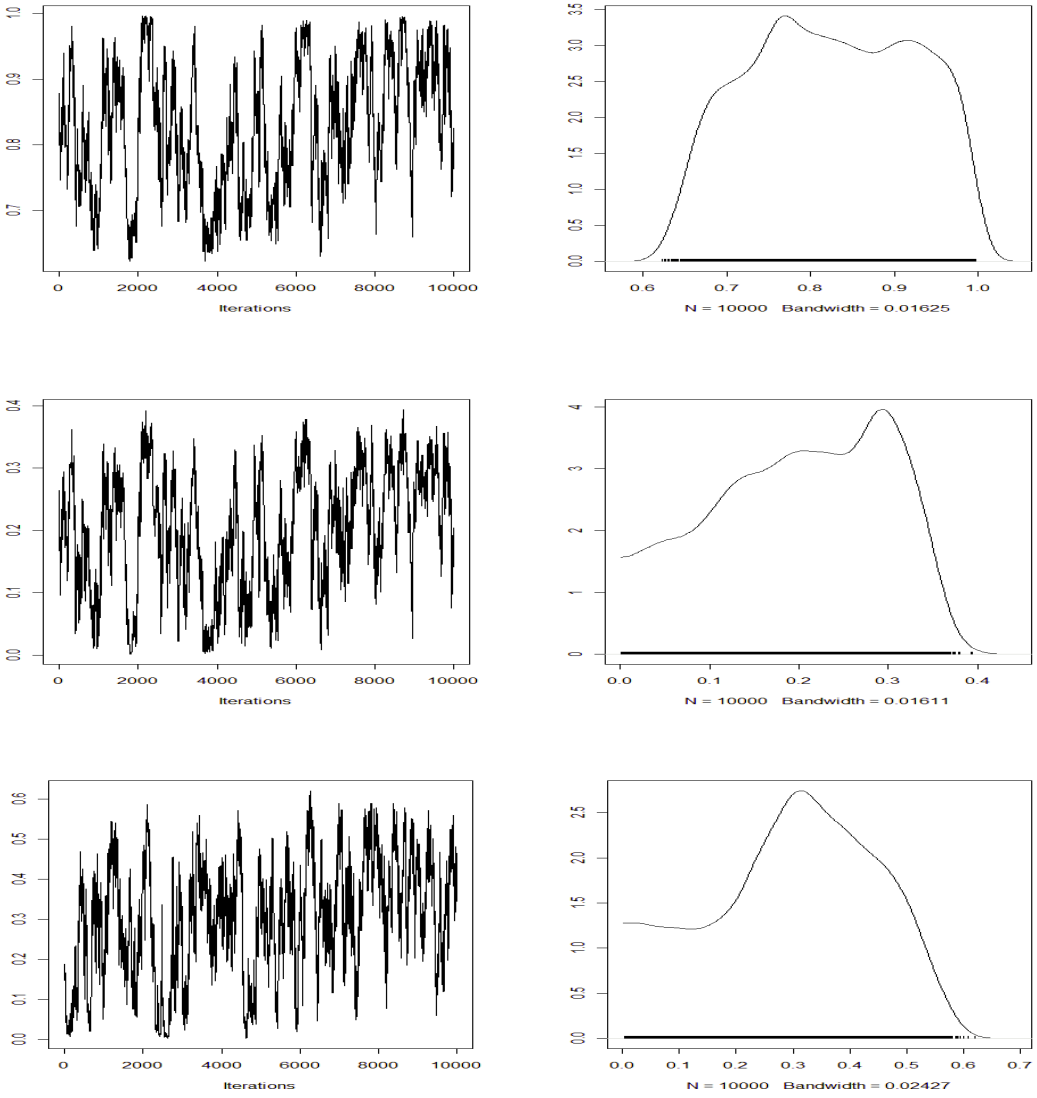
What about an *extrinsically* non-identifiable parameter? Recall from section F.1 that real parameter 14 (second recapture probability for male Dippers) was estimated at or near the boundary. The data cloning approach led us to conclude that this parameter was in fact correctly estimated as being at the boundary. What does the MCMC approach show?



We see (above) that the density plot is strongly left-skewed, as the information in the data 'piles up'

against the 1.0 boundary. Since extrinsic non-identifiability is almost always reflected in a parameter being estimated at either the [0, 1] boundary, it may be unlikely that the MCMC approach will be of much use in diagnosing extrinsic non-identifiability.

Further, even for intrinsically non-identifiable parameters, the density plot can be difficult to interpret, and may in fact bear no particular resemblance to the prior distribution (i.e., does not look particular similar to the default 'flat' uniform prior). For example, consider the robust design analysis presented in section F.3. In the absence of certain structural constraints, we expect that the final survival parameter S and the final two γ parameters will be intrinsically confounded (for the example data set, this involved real parameters 4, 8 and 11). This intrinsic confounding was demonstrated clearly using the data cloning approach. But, have a look at the trace and density plots for any of these 3 parameters (below, for parameters 4, 8 and 11, respectively):



While these plots are clearly 'strange looking', that is a somewhat 'subjective' assessment, and not based on a clear resemblance of the posterior to the prior. However, even such a 'subjective assessment may be enough to warrant further consideration of the identifiability of those parameters.

end sidebar

F.6. Summary

In this appendix, we've briefly introduced a convenient, fairly straightforward method for numerically assessing whether a parameter is 'estimable' or not. This method, based on 'data cloning', appears to work well, especially for 'standard' parameters that are $[0, 1]$ bounded (e.g., φ and p in a CJS model) – both for parameters that may be confounded (and thus not separately identifiable) because of the structure of the model, and for parameters that are poorly estimated because of inadequate data, or because they are near the $[0, 1]$ boundary.

F.7. Literature Cited

Gimenez O., A. Viallefont, R. Choquet, E. A. Catchpole, and B. J. T. Morgan. 2004. Methods for investigating parameter redundancy. *Animal Biodiversity and Conservation*, **27**, 561-572.

Gimenez O., B. J. T. Morgan, and S. P. Brooks. 2009. Weak identifiability in models for mark-recapture-recovery data. In *Modeling Demographic Processes in Marked Populations*, D. L. Thomson, E. G. Cooch and M. J. Conroy, eds. Springer, New York, New York, USA, pp. 1055-1067.

Hunter, C. M., and H. Caswell. 2009. Rank and redundancy of multistate mark-recapture models for seabird populations with unobservable states. In *Modeling Demographic Processes in Marked Populations*, D. L. Thomson, E. G. Cooch and M. J. Conroy, eds. Springer, New York, New York, USA, pp. 324-333.

Lele, S. R., B. Dennis, and F. Lutscher. 2007. Data cloning: easy maximum likelihood estimation for complex ecological models using Bayesian Markov chain Monte Carlo methods. *Ecology Letters*, **10**, 551-563.

Lele, S. R., K. Nadeem, and B. Schmuland. 2010. Estimability and likelihood inference for generalized linear mixed models using data cloning. *Journal of the American Statistical Association*, **105**, 1617-1625.

APPENDIX G

The 'data bootstrap. . .'

The following introductory material was adopted liberally from Bishop *et al.* (2008), the main published paper describing the data bootstrap in Program **MARK**. The data bootstrap is also commonly referred to as the nonparametric bootstrap in the statistical literature. Efron and Tibshirani (1993) and Chernick (1999) provide extensive descriptions and advice on the use of the bootstrap method.

Independence of sample units is required in survival analyses to obtain appropriate estimates of sampling variance. The independence assumption is likely violated when sample units include monogamous mates or siblings. For example, adult pairs of Canada geese (*Branta canadensis*) typically mate for life and, therefore, do not behave independently of one another (Anderson *et al.* 1994, Sheaffer *et al.* 2004). Littermates of numerous species access the same nutrient supply and are often subjected to the same mortality risks. Some degree of biological dependence likely exists among individuals that use the same resources in time and space.

When the independence assumption is violated, survival point estimates are typically unbiased, but sample variances are underestimated (Wedderburn 1974, Cox 1983, Firth 1987, Breslow 1990, Schmutz *et al.* 1995). This condition is referred to as overdispersion or extra- binomial variation. Sample data will be overdispersed if siblings each die, or each survive, more often than expected under the assumption of independence. An overdispersion parameter, or variance inflation factor, c, is required to correct for overdispersed data, thereby facilitating appropriate inference using quasi-likelihood theory (Wedderburn 1974, Burnham and Anderson 2002; see Chapter 4 and Chapter 5). The simplest approach for estimating c requires only the standard goodness-of-fit (GOF) test statistic for the global (i.e., most parameterized) model and its degrees of freedom (Cox and Snell 1989). The global model is needed to avoid mistaking model structure variation (i.e., lack of model fit) for overdispersion (Burnham and Anderson 2002). Chi-square GOF approaches have been used to address sibling dependence for a variety of species (Winterstein 1992, Gaillard *et al.* 1998, Schwartz *et al.* 2006, Wiens *et al.* 2006). Unfortunately, these approaches fail to provide an appropriate estimate of c under many circumstances because of inadequate sample sizes. Instead, more sophisticated numerical approaches are necessary (Schmutz *et al.* 1995; Bishop *et al*. 2008). The data bootstrap we introduce here provides a method to evaluate whether theoretical variance estimates are valid when the individuals are considered statistically independent, or whether variance inflation procedures are required to account for dependence among (say) siblings.

The simulation procedure in **MARK** allows bootstrapping the encounter histories data to generate bootstrap replication where an individual covariate is used to identify the blocks of data (litters, family groups, etc.) that are resampled with the bootstrap procedure. To bootstrap encounter histories, you select the '**`Bootstrap Data`**' option from the '**`Simulation`**' menu choices. A potential trap in using this procedure is that the encounter history file should be set up to have each history represent the

appropriate number of animals. Typically, this would be 1 animal, but as the example below shows, such may not be the case. Encounter history files with encounter histories pooled so that the count frequency is > 1 will work correctly with this procedure because the histories are unpooled for sampling. Be careful about the group structure of your data – encounter histories *within* groups are resampled, never *across* groups.

Bootstrapping of individual encounter histories will not provide any additional information from the usual analyses. That is, resampling individual encounter histories produces the same asymptotic estimates and standard errors as the default maximum likelihood approach in **MARK**. Where bootstrapping the encounter histories is most useful is when there are dependencies across animals. As described above, the most typical example of such data are marking of young that are known to be siblings. So, all the young in a litter or nest are marked. When the encounters of these animals are treated as if they are independent, the estimates will be generally unbiased, but the standard errors will be too small because the sampling unit is really the litter or nest, and not the individual animal. So, in the maximum likelihood estimation assuming independence of individuals, the sample size is assumed to be larger than it really should be.

G.1. Empirical example

To bootstrap litter or nest data such as described above, include an individual covariate that defines the nest or litter. This variable can be continuous, such as nest number 1 to the number of nests, or litter number. The main criterion is that each litter or nest is *uniquely identified* with this covariate. This covariate is then used to bootstrap the encounter histories, such that the nests or litters are resampled, instead of the individual encounter histories.

The example we use here to illustrate the data bootstrap analysis is the neonatal fawn survival study described in Bishop *et al.* (2008). Adult female deer commonly produce twins and occasionally triplets. Siblings should not be assumed to have independent fates, because they have the same dam, share $\approx$ 0.25 of their genome, and are exposed to nearly identical environmental conditions postpartum. Maternal body condition and disease status, as well as predation, are mechanisms that could cause dependence among sibling fates. However, siblings are commonly radio-collared in neonatal survival studies (Hamlin *et al.* 1984, Whittaker and Lindzey 1999, Carstensen *et al.* 2003, Jarnemo and Liberg 2005, Bishop *et al.* 2008). There are advantages to marking siblings, but the potential dependence of survival outcomes should be addressed. Sample unit dependence is also encountered in measurements of fetal survival, which can be made in free-ranging deer populations with the aid of ultrasonography and vaginal implant transmitters (Bishop *et al.* 2007). Inclusion of siblings is required when estimating fetal survival because the fetus sample is obtained from *in utero* counts, and each of an adult female's fetuses must be counted without error. Twin or triplet fetuses cannot be individually marked *in utero*, and, therefore, one fetus cannot be randomly chosen to include in the sample and others disregarded.

All fetuses and neonates used in the study described in Bishop *et al*. (2008) were offspring of free-ranging, radio-collared adult female mule deer. We administered a nutrition enhancement treatment to half of the adult females during winter and early spring to meet research objectives described elsewhere (Bishop *et al.* 2009). No nutrition treatment was applied to the other half of the adult females, hereafter control deer. Neonatal fawns were followed for 182 days (26 weeks). The encounter histories were entered as 182 occasions for 6 groups. The top of the input file (`deer-bootstrap.inp`), plus the first 3 encounter histories are shown on the next page, where the encounter histories wrap across lines because of their length (264 characters).

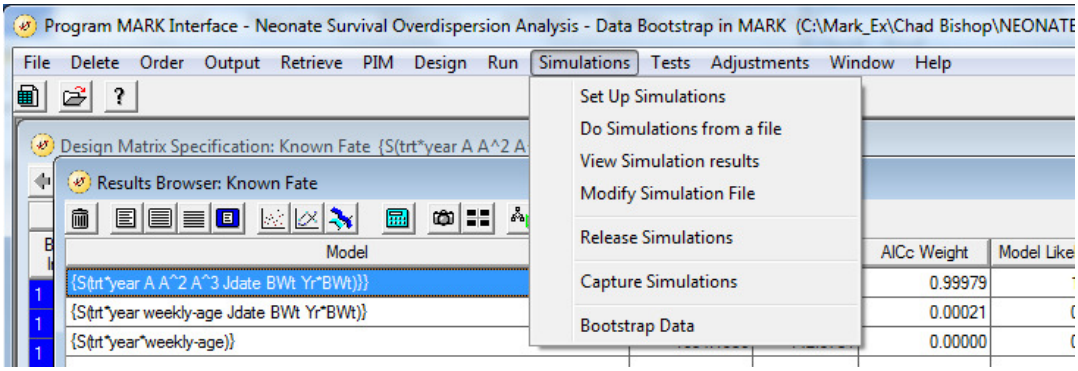
```
/* Mule Deer Neonates, Uncompahgre Plateau, 2002-2004
   Known Fates
   occasions=182 groups=6
   occasions represent daily intervals and are expressed in terms of age (i.e., days since birth)
   glabel(1)=Control 2002
   glabel(2)=Treatment 2002
   glabel(3)=Control 2003
   glabel(4)=Treatment 2003
   glabel(5)=Control 2004
   glabel(6)=Treatment 2004
   covariate1=Date of capture expressed as the difference between capture date and earliest
   capture date of the year
   covariate2=Litter ID
   covariate3=Estimated Birth Weight
   covariate4=Estimated Birth Hind Foot Length           */
    0000101010101010101100000000000000000000000000000000000000000000
    0000000000000000000000000000000000000000000000000000000000000000
    0000000000000000000000000000000000000000000000000000000000000000
    0000000000000000000000000000000000000000000000000000000000000000
    0000000000000000000000000000000000000000000000000000000000000000
    0000000000000000000000000000000000000000000000000000000000000000  1 0 0 0 0 0 4 1 2.4335 23.4162 ;
    0000001010101010101010101010101010101010101010101010101010101010101
    0101010101010101010101010101010101010101010101010101010101010101010
    1010101010101010101010101010101010101010101010101010101010101010101
    0101010101010101010101010101010101010101010101010101010101010101010
    1010101010101010101010101010101010101010101010101010101010101010101
    0101010101010101010101010101010101010  1 0 0 0 0 0 4 1 2.9613 26.021 ;
    0000000010101010101010101010101010101010101010101010101010101010101
    0101010101010101010101010101010101010101010101010101010101010101010
    1010101010101010101010101010101010101010101010101010101010101010101
    0101010101010101010101010101010101010101010101010101010101010101010
    1010101010101010101010101010101010101010101010101010101010101010101
    0101010101010101010101010101010101010  1 0 0 0 0 0 18 2 3.7237 26.357 ;
```

Survival was estimated on a weekly basis by creating PIMs with daily survival constant within the 26 weeks. Thus, 26 real parameters are estimated for each of the 6 groups, giving 156 real parameters.

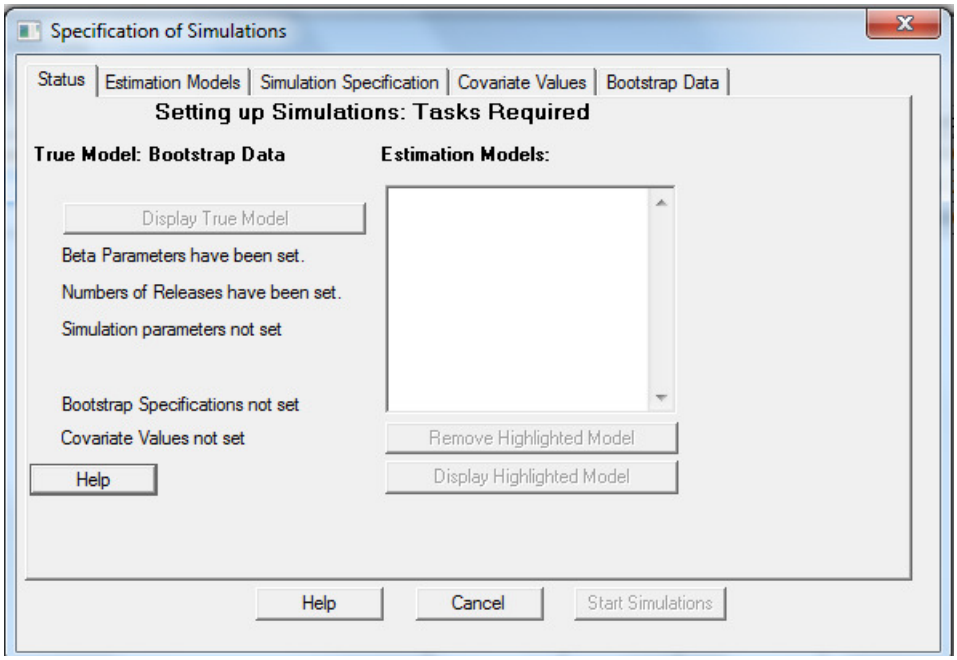
Four individual covariates are specified as `Jdate` = date of capture expressed as the difference between capture date and earliest capture date of the year, `LitID` = Litter ID (to be used to perform the bootstrap resampling, `BWt` = Estimated Birth Weight, and `BHFt` = Estimated Birth Hind Foot Length. So, the first 2 encounter histories above correspond to `LitId` = 1, and the third encounter history is for `LitID` = 2.

To save you from having to build some of the models described in Bishop *et al*. (2008), open up `deer-bootstrap.dbf` (making sure that `deer-bootstrap.fpt` is in the same directory).

To run the data bootstrap, select the '**Bootstrap Data**' menu choice from the '**Simulation**' menu in the Results Browser.

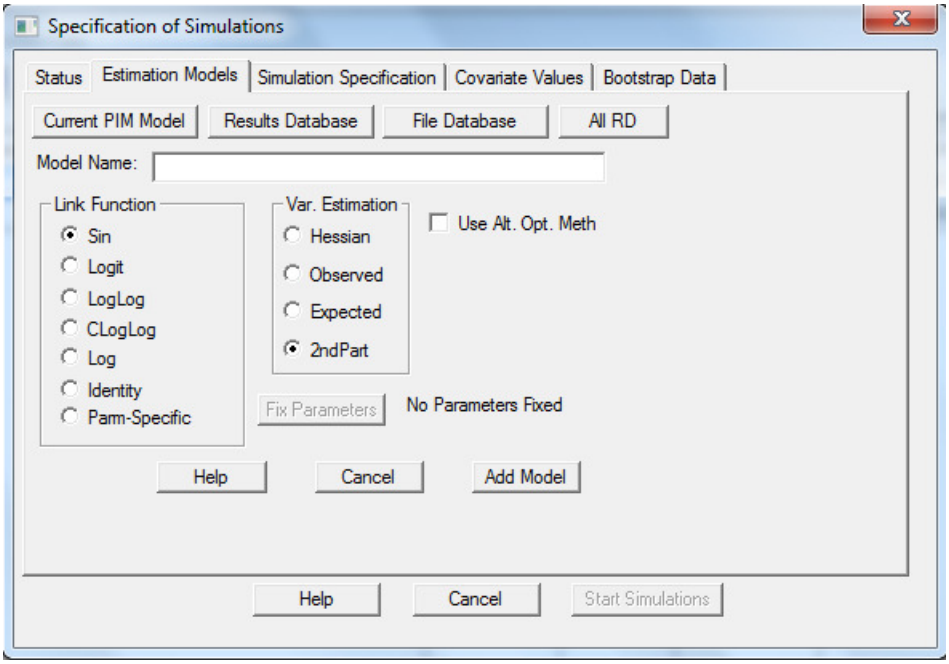


Now you need to provide the information requested in the 5 tab windows (if you are new to running simulations in **MARK**, it might be useful to review Appendix A before continuing – here we only provide a summary of the simulation capabilities in **MARK**). The '**Status**' tab just summarizes your progress in providing the necessary information, and is the tab window shown first (below). You are returned to the '**Status**' tab as you move through the other 4 tabs to view your progress.



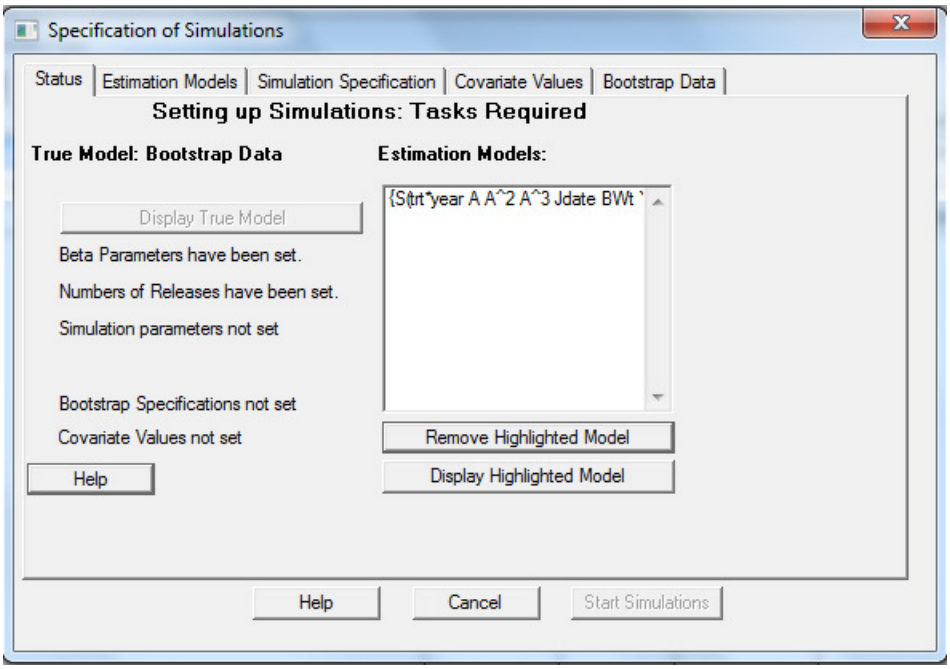
The '**Status**' window for a data bootstrap analysis, with all the default values shown. Notice that none of the 5 messages displayed on the left side of the screen indicate you have specified these values. After you have completed the entries in the 4 right tab windows, all of these messages will indicate you have set up the bootstrap analysis. At that time, the '**Start Simulations**' button will become live for you to proceed with the numerical analysis.

To start, select 1 or more models to estimate from the '**Estimation Models**' tab (below).

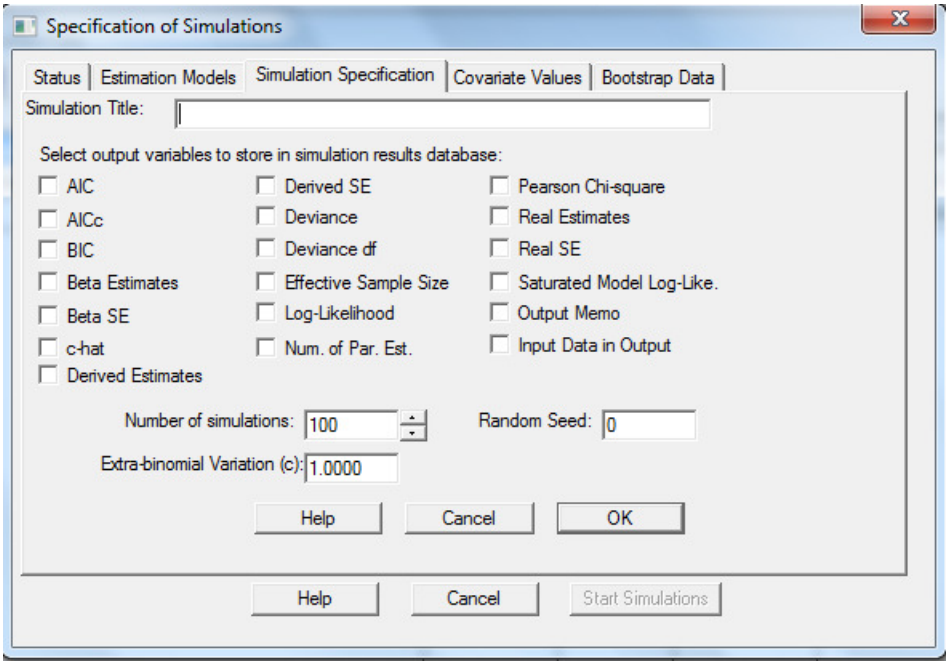


You can select models with the 4 push buttons across the top of the window. The '**Current PIM Model**' button selects the model you currently have constructed in the PIMs (and the associated design matrix if you have created one). The '**Results Database**' push button retrieves the currently highlighted model from the Results Browser. The '**File Database**' button retrieves a model from another **MARK** file - you will be given a list of the models in the file once you have selected it with the usual Windows file selection windows. For each of these buttons, you have to manually add the model to the list of models for which you want estimates with the '**Add Model**' button in the lower center portion of the tab window. The fourth button for selection models is the '**All RD**' button, which automatically adds all the models in the results database, and is useful for when you want to do model selection using the bootstrap results. You do not need to use the '**Add Model**' button with the '**All RD**' button.

For this example, we'll assume that the '**Current PIM Model**' button is selected, and that the highlighted model in the results browser is '`S(trt*year A Â 3 Jdate BWt Yr*BWt)`'. The model name is filled in, and the link function is changed to '**Logit**' because this model uses a logit link with the design matrix. After the model is added with the '**Add Model**' button, the '**Status**' tab now looks like that shown in at the top of the next page. The '**Use Alt. Opt. Meth**' check box allows you to specify that the simulated annealing optimization method be used (which typically takes 10× longer to converge) instead of the default optimization method. Also notice that now the '**Remove Highlighted Model**' and '**Display Highlighted Model**' buttons are active because there is an active estimation model displayed. You can continue to add models to the '**Estimation Model**' list by going back to the estimation models tab. Likewise, the messages show that the '**Beta Parameters**' have been set, and the '**Number of Releases**' have been set. These values are obtained from the data for the estimation model.

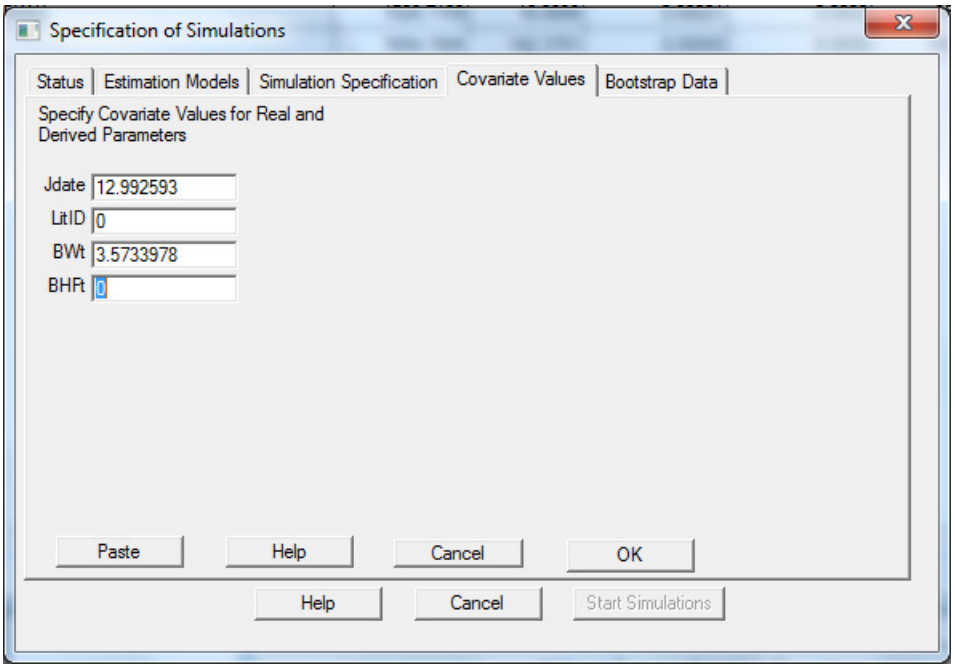


Next, the parameters of the simulation that you want to save in the simulation results database are selected with the Simulation Specification tab (Figure 4). For this example, we are most interested in the derived parameters of the survival estimate across the 6-month interval, so we would check the '**`Derived Estimates`**' and the **`Derived SE`**' boxes. In addition, we would want to specify more than the 100 simulations, so would likely specify 1,000, or as Bishop *et al.* (2008) did, 10,000. Chernick (1999) suggests a minimum of 5,000 given the speed of today's computers.



The '**Extra-binomial Variation (c)**' box is not useful for this example, but provides a method to add overdispersion to the existing data. That is, the encounter histories are duplicated as necessary to provide data representing the value of *c* specified. The '**Random Seed**' box allows you to specify a random integer, so that you can repeat the analysis at some future time with exactly the same bootstrap samples. The value of zero that is the default uses the computer's clock to generate a random value to start the resampling process. The '**Simulation Title**' currently does nothing, so can be left blank. When the appropriate values have been entered, push the '**OK**' button to proceed, which returns you to the Status tab window. The '**Status**' tab window now shows that the '**Simulation Parameters**' have been set.

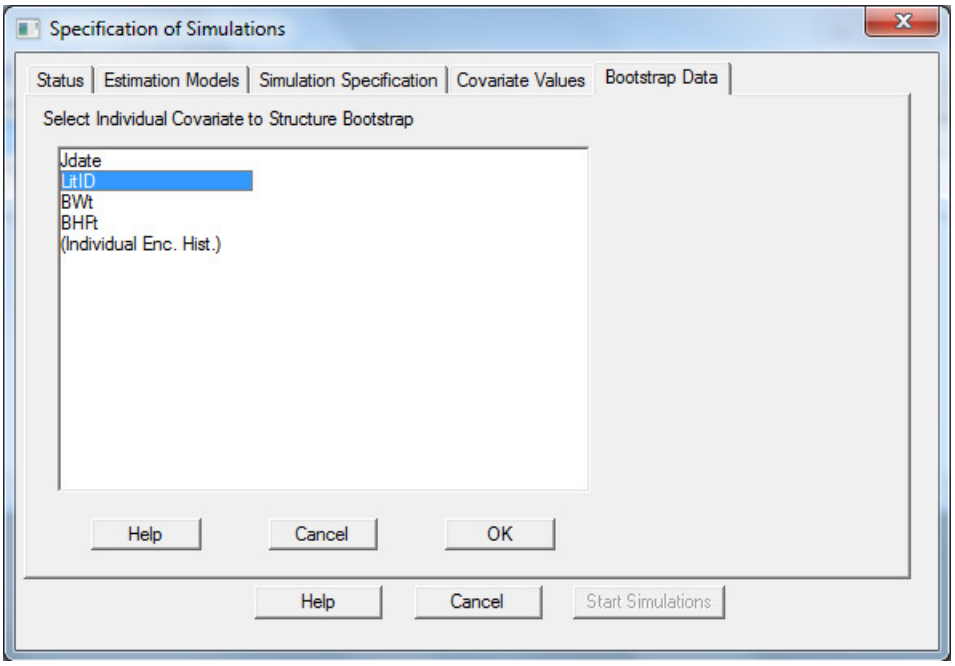
Because the model being estimated uses values of the individual covariates `Jdate` (Julian date of birth) and `BWt` (birth weight) in the design matrix, we need to specify values to use for the bootstrap resampling with the '**Covariate Values**' tab window (below). For the example demonstrated here, these are the values used to compute the real parameter values (26 weekly survival rates for each of 6 groups), and thus the survival rate across the entire 26-week interval.



The '**Covariate Values**' tab window specifies the individual covariate values to use for computing real and derived parameter values. The default values are all zeros, but in the above screen, the mean values for Julian date (`Jdate`) and birth weight (`BWt`) have been entered because these 2 individual covariates are used to compute the real parameters in the model being estimated.

After the '**OK**' button is pushed, you are returned to the '**Status**' tab window, which now specifies that the '**Covariate Values**' have been set.

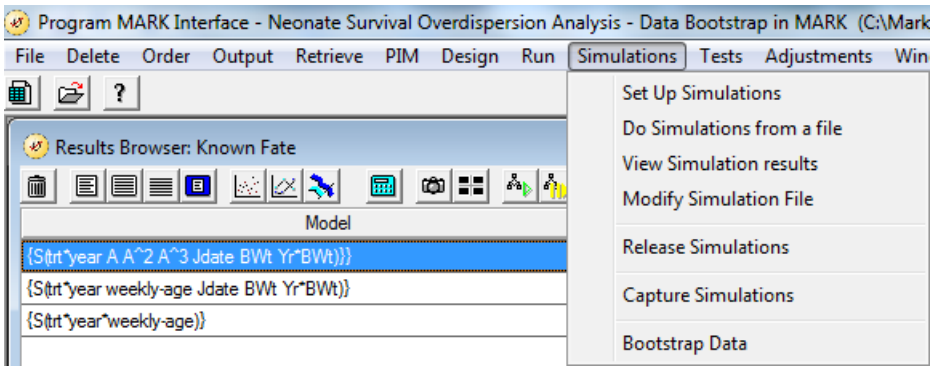
The last tab window is the '**Bootstrap Data**' tab, which is where you select the individual covariate to be used to resample the data. In our example, the LitID covariate (litter ID) is selected because this covariate identifies the individual litters. Although the LitId covariate is continuous across all 6 groups, resampling only occurs *within* groups to form the resampled dataset for each group.



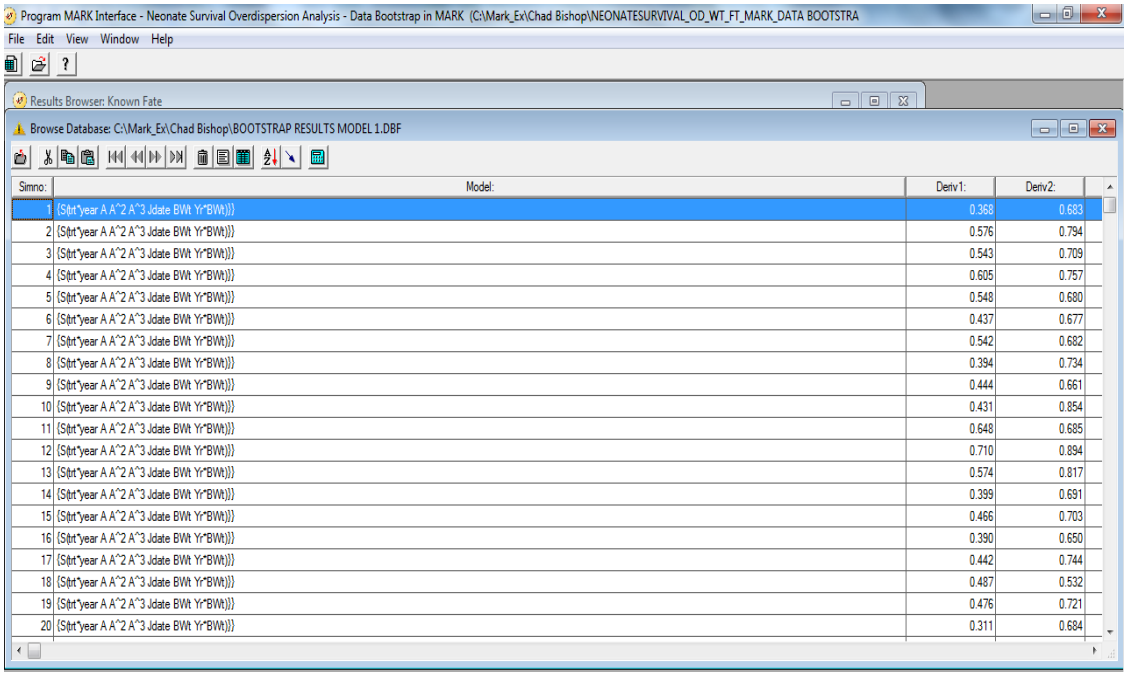
The '**(Individual Enc. Hist.)**' value is also displayed in the list of individual covariates, but is not actually a covariate. Rather, this option can be used to resample the individual encounter histories. Care must be taken that each encounter history represents the same number of animals, or else the resampling will be a sample biased towards encounter histories with the most animals.

Clicking on the '**OK**' button returns you to the '**Status**' tab window, which now shows that '**Bootstrap Specifications**' have been set. In addition, the '**Start Simulations**' push button at the lower right corner of the window is 'live', meaning that all the information needed to perform the bootstrap sampling process has been provided and you can proceed to the numerical work.

When you click on the '**Start Simulations**' button, you will be asked to specify a file to store the bootstrap results into. The default file name is SimResults.DBF - so you should probably provide a more informative name. We'll use Bootstrap Results Model1.DBF. For our example here, the values of the 6 derived parameters and their standard errors will be stored for 1000 bootstrap resamples. When the simulations finish, you can view the results with the '**View Simulation Results**' menu choice (shown at the top of the next page). This menu choice takes you to a Windows dialog to select the file where simulations were placed, in this case 'Bootstrap Results Model1.DBF'.



Select this file, and you get the following window.



The column widths can be changed by placing the cursor on the column boundary and dragging. You will find that you have 6 columns of estimates of the 6 derived parameters (survival rate across the entire 26 weeks for each of the 6 groups) labeled `DERIV1` to `DERIV6`, and you'll also have 6 columns with the standard errors of each of these estimates labeled `SED1` to `SED6`.

Clicking on the right-most button on this window (the '**Calculator**' button) will calculate some basic summary statistics and put them into a notepad window.

```
                Statistical Summary of Numerical Variables
                     (Number of Observations = 1000)

                                                                  95\% Confidence Interval
  Variable            Mean   Standard Dev.  Standard Error           Lower           Upper
---------- ---------------  --------------  -------------- --------------- ---------------
    DERIV1           0.526          0.1313          0.0042           0.518           0.534
    DERIV2           0.738          0.1099          0.0035           0.732           0.745
    DERIV3           0.471          0.0867          0.0027           0.465           0.476
    DERIV4           0.613          0.0753          0.0024           0.608           0.618
    DERIV5           0.428          0.0749          0.0024           0.423           0.433
    DERIV6           0.405          0.0757          0.0024           0.400           0.409
      SED1           0.112          0.0094          0.0003           0.112           0.113
      SED2           0.090          0.0185          0.0006           0.088           0.091
      SED3           0.077          0.0041          0.0001           0.076           0.077
      SED4           0.069          0.0036          0.0001           0.069           0.069
      SED5           0.073          0.0041          0.0001           0.073           0.073
      SED6           0.064          0.0041          0.0001           0.064           0.064
```

This summary is just a quick look at your results, often enough to verify that everything is working okay. However, to perform a more intensive analysis, you will need to import the DBF file into a statistical package or into Excel. As an example, the following SAS code will import the DBF file into SAS.

```
libname library V9 '.';
title 'Analyze Bootstrap Data from Program MARK';
filename BootData 'BOOTSTRAP RESULTS MODEL 1.DBF';
proc dbf db4=BootData out=BootstrapData;
proc means;
run;
```

For our example here, the necessary information to compute overdispersion is available. For `DERIV1`, the standard deviation of the 1000 bootstraps is 0.1313. From the maximum likelihood analysis stored in the Results Browser, the standard error of this parameter is 0.1139. So an estimate of c for `DERIV1` is $(0.1313)^2/(0.1139)^2 = 1.329$, which is in close agreement with the published result of 1.297 in Table 4 of Bishop *et al.* (2008). Because the bootstrap procedure is a random resampling process, you will never generate the exact same results unless the same random seed is specified. Further note that the value of the standard error used above does not match exactly the value of 0.1143 reported by Bishop *et al.* (2008) - probably because of a slightly improved optimization procedure currently in **MARK** over the procedure used by them.

G.2. Extensions

The data bootstrap procedure can also be used for bootstrapping the model selection process. To do this, you must compute estimates for all the models with each of the bootstrap resamples. Note that the

Status tab window (Figure 3) allows more than one estimation model, i.e., you can keep going back to the Estimation tab window (Figure 2) to add additional models. Or you can use the All RD button in the Estimation tab window to add all of the models in your results browser at one time.

Once the bootstrap resampling is completed, you can then evaluate model selection results, although these more complex analyses will require analyzing the bootstrap results in a statistical package. For example the proportion of times a model was selected as the minimum AIC_c model in the bootstrap resamples could be taken as the model's weight. Buckland *et al.* (1997) suggest more sophisticated methods of model selection based on bootstrap resampling.

G.3. Limitations

The bootstrap data procedure only works with encounter history files containing encounter histories, and will not work with the summary input that is allowed for band recovery data or known fate data. You have to convert these formats to the corresponding encounter histories. One trick to do this is to use the residuals output file in Notepad, where the data are given as encounter histories.

One issue that users of the bootstrap data procedure should be aware of is how the frequency counts for encounter histories are handled. These values are maintained with the encounter history, so are not changed by the bootstrap procedure. So, if for some reason you want to bootstrap individual encounter histories, you may want to specify each animal separately, each with a frequency of 1. With the use of the individual covariate to specify the resampling structure, the frequencies can be > 1 and make perfect sense. As an example, consider an input file for known fate data with 1 occasion. An individual covariate of Litter, with values of 1 through 4, is also included in the encounter histories file.

```
11 1 1;
10 2 1;
11 2 2;
10 2 2;
11 1 3;
10 3 4;
```

The above input file shows 4 litters of size 3, 4, 1, and 3, respectively. For this reason, the count frequency for each encounter history can exceed 1, because the litter identifier provides the blocking to bootstrap the encounter histories. Bootstrapping would be performed by resampling from the 4 litters if the '**Litter individual covariate**' was specified in the '**Bootstrap Data**' tab window. If the '**Indiv. Enc. Hist.**' choice was selected (p. G-8), the preceding file would still generate useful bootstrap estimates because the encounter histories with counts > 1 would be resampled according to the encounter history count.

Groups are kept separate in the bootstrap resampling process. Thus, the number of litters is kept constant for each group in the resampled data. Even though your model being estimated might ignore or combine groups, the bootstrap resampling will not ignore these groups. The number of unique blocks of histories within each group will be sampled within that group, and never across attribute groups. Also, for multi-state data, the bootstrap resampler does not distinguish between initial states, so that the conditioning on number of animals starting in each state is lost.

G.4. Literature Cited

Anderson, D. R., K. P. Burnham, and G. C. White. 1994. AIC model selection in overdispersed capture-recapture data. *Ecology*, **75**, 1780-1793.

Bishop, C. J., D. J. Freddy, G. C. White, B. E. Watkins, T. R. Stephenson, and L. L. Wolfe. 2007. Using vaginal implant transmitters to aid in capture of mule deer neonates. *Journal of Wildlife Management*, **71**, 945-954.

Bishop, C. J., G. C. White, and P. M. Lukacs. 2008. Evaluating dependence among mule deer siblings in fetal and neonatal survival analyses. *Journal of Wildlife Management*, **72**, 1085-1093.

Bishop, C. J., G. C. White, D. J. Freddy, B. E. Watkins, and T. R. Stephenson. 2009. Effect of enhanced nutrition on mule deer population rate of change. *Wildlife Monographs*, **172**.

Breslow, N. 1990. Tests of hypotheses in overdispersed Poisson regression and other quasi-likelihood models. *Journal of the American Statistical Association*, **85**, 565-571.

Buckland, S. T., K. P. Burnham, and N. H. Augustin. 1997. Model selection: An integral part of inference. *Biometrics*, **53**, 603-618.

Burnham, K. P., and D. R. Anderson. 2002. *Model selection and multimodel inference: a practical information-theoretic approach. Second edition*. Springer-Verlag, New York, New York, USA.

Carstensen, M., G. D. DelGiudice, and B. A. Sampson. 2003. Using doe behavior and vaginal-implant transmitters to capture neonate whitetailed deer in north-central Minnesota. *Wildlife Society Bulletin*, **31**, 634-641.

Chernick, M. R. 1999. *Bootstrap Methods A Practitioner's Guide*. Wiley Series in Probability and Statistics, New York, New York, USA.

Cox, D. R. 1983. Some remarks on overdispersion. *Biometrika*, **70**, 269-274.

Cox, D. R., and E. J. Snell. 1989. *Analysis of binary data. Second edition*. Monographs on Statistics and Applied Probability 32. Chapman and Hall/CRC, New York, New York, USA.

Efron, B., and R. J. Tibshirani. 1993. *An Introduction to the Bootstrap*. Monographs on Statistics and Applied Probability 57, Chapman & Hall, New York, New York, USA.

Firth, D. 1987. On the efficiency of quasi-likelihood estimation. *Biometrika*, **74**, 233-245.

Gaillard, J-M., R. Andersen, D. Delorme, and J. D. C. Linnell. 1998. Family effects on growth and survival of juvenile roe deer. *Ecology*, **79**, 2878-2889.

Hamlin, K. L., S. J. Riley, D. Pyrah, A. R. Dood, and R. J. Mackie. 1984. Relationships among mule deer fawn mortality, coyotes, and alternate prey species during summer. *Journal of Wildlife Management*, **48**, 489-499.

Jarnemo, A., and O. Liberg. 2005. Red fox removal and roe deer fawn survival – a 14-year study. *Journal of Wildlife Management*, **69**, 1090-1098.

Schmutz, J. A., D. H. Ward, J. S. Sedinger, and E. A. Rexstad. 1995. Survival estimation and the effects of dependency among animals. *Journal of Applied Statistics*, **22**, 673-681.

Schwartz, C. C., M. A. Haroldson, and G. C. White. 2006. Survival of cub and yearling grizzly bears in the Greater Yellowstone Ecosystem, 1983-2001. Pages 25-31 in C. C. Schwartz, M. A. Haroldson, G. C. White, R. B. Harris, S. Cherry, K. A. Keating, D. Moody, and C. Servheen, 2006. Temporal, spatial, and environmental influences on the demographics of grizzly bears in the Greater Yellowstone Ecosystem. *Wildlife Monographs*, **161**.

Sheaffer, S. E., D. H. Rusch, D. D. Humburg, J. S. Lawrence, G. G. Zenner, M. M. Gillespie, F. D. Caswell, S. Wilds, and S. C. Yaich. 2004. Survival, movements, and harvest of eastern prairie population Canada geese. *Wildlife Monographs*, **156**.

Wedderburn, R. W. M. 1974. Quasi-likelihood functions, generalized linear models, and the Gauss-Newton method. *Biometrika*, **61**, 439-447.

Whittaker, D. G., and F. G. Lindzey. 1999. Effect of coyote predation on early fawn survival in sympatric deer species. *Wildlife Society Bulletin*, **27**, 256-262.

Wiens, J. D., B. R. Noon, and R. T. Reynolds. 2006. Post-fledging survival of northern goshawks: the importance of prey abundance, weather, and dispersal. *Ecological Applications*, **16**, 406-418.

Winterstein, S. R. 1992. Chi-square tests for intrabrood independence when using the Mayfield method. *Journal of Wildlife Management*, **56**, 398-402.

Index

— *A* —

abundance estimation, *see* closed population models
age models
    age vs cohort vs time, **7**:2
    design matrix
        marked as young & adult, **7**:26
    goodness of fit, **7**:42
    introduction, **7**:2
    linear constraints, **7**:14
        >2 age classes, **7**:22
        interaction terms, **7**:15, **7**:22
    model averaging, **7**:47
    parameter counting, **7**:13
    PIM structure, **7**:3-6
    time since marking (TSM), **7**:32
    transience example, **7**:32
AIC
    AIC weights
        evidence ratios, **4**:39
    derivation, **4**:30
    model weights
        derivation, **4**:37
        motivation, **4**:37
    random effects models, **D**:23
    stepwise regression, **4**:57
    weights
        model likelihood, **4**:39

— *B* —

batch-release, **5**:11

— *C* —

changing data types, **14**:28
CJS model
    as multi-state problem, **10**:28
closed population models
    confidence intervals
        profile likelihood, **14**:39
    definition of closure, **14**:5
    goodness of fit, **14**:27
    heterogeneity models, **14**:20
        π, **14**:25
    misidentification, **14**:26
    mixture models, **14**:20
    model averaging, **14**:27
        changing data types, **14**:28
        confidence intervals, **14**:35
    model notation, **14**:10
    model types, **14**:6
        Huggins models, **14**:4
        likelihoods, **14**:6
    negative AIC, **14**:12
    parameter estimability, **14**:40
    problem of heterogeneity, **14**:25
    removal models, **14**:14
cohort
    definition, **3**:10
cohort models
    design matrix, **7**:44
    introduction, **7**:42
    model averaging, **7**:47
combined dead recovery-live encounter models, **9**:1
    Barker models, **9**:11
        movement, **9**:12
    marked as young only, **9**:10
    multi-state extensions, **9**:13
    parameter identifiability, **9**:15
    probability structure, **9**:3
confidence limits
    back-transformation, **4**:50, **B**:17, **B**:19
counting parameters
    general, **4**:61
    technical
        estimability, **4**:66

— *D* —

data bootstrap
    extensions, **G**:10
    limitation, **G**:11
    non-independence, **G**:1
data cloning, **F**:1
data formatting

editors, **2**:2
dead recovery model
combined with live encounters, **9**:1
dead recovery models
as multi-state problem, **10**:31
Brownie parameterization, **8**:1
marked as young and adults, **8**:12
marked as young only, **8**:10
parameter counting, **8**:7
probability structure, **8**:2
enconter history, **2**:7
euphimisms for 'killed', **8**:1
goodness of fit, **8**:25
Program ESTIMATE, **8**:27
number marked unknown
BOU parameterization, **8**:23
PIMs, **8**:4
Seber (S and r) parameterization, **8**:16
probability structure, **8**:16
Delta method, **11**:14, **B**:1
effect size
standard error, **6**:66
model averaging, **B**:27
multivariate transformations, **B**:10
random variables
moments, **B**:1
reporting rate, **B**:14
SE, **B**:15, **B**:17, **B**:20
single variable transformations, **B**:7
Taylor series expansions, **B**:5, **F**:17
transformations, **B**:3
variance of a product, **B**:11
violation of assumptions, **B**:7
design matrices
column functions, **11**:20, **17**:18
fulle vs. reduced vs. identity, **6**:21
functions, **11**:20
design matrix
see linear models, **6**:7
subset models, **6**:33

— ***E*** —

encounter
history, **2**:1
encounter histories
known-fate data, **16**:5
encounter history
> 1 classification factor, **2**:4
Brownie recovery, **2**:7
formats, **2**:2
removing individuals, **2**:4
summary list, **2**:5
individual covariates, **2**:8
scaling, **2**:9
interpretation, **1**:3
multiple groups, **2**:3
nest survival, **17**:4-8
cell probabilities, **17**:7
probability expression, **1**:4
example
cost of breeding, **10**:6
European dipper
flood, **6**:12
European dippers
males, **3**:1-16
metapopulation, **10**:17
transience, **7**:32
example data files, ii

— ***G*** —

goodness of fit
$\hat{c}$
$\hat{c} < 1$?, **5**:6
introduction, **5**:2
QAIC, **5**:5
rules-of-thumb, **5**:37
age and TSM models, **7**:42
bootstrap, **5**:24-28
closed population models, **14**:27
data bootstrap, **G**:1
dead recovery models, **8**:25
Program ESTIMATE, **8**:27
deriving $\hat{c}$
RELEASE, **5**:21
U-CARE, **5**:21
deviance residual plots, **5**:33
deviance residuals plot, **7**:34
full m-array, **5**:12
general model
time-dependence, **5**:22
individual covariates, **11**:46
lack of fit
extra-binomial variation, **5**:35
model structure, **5**:32
median $\hat{c}$, **5**:28-30
versus RELEASE, bootstrap, **5**:30
multi-state models, **10**:46
median $\hat{c}$, **10**:48
U-CARE, **10**:46
nest survival, **17**:19
problems estimating $\hat{c}$, **5**:31
Program MARK
simulation, **5**:37
Program RELEASE, **5**:7
full m-array, **5**:13
Test 2, **5**:9

Test 3, **5**:10
use of component tests, **5**:22
Program Release, **5**:7
U-CARE, **5**:17
reduced m-array, **5**:9
robust design, **15**:47
saturated model, **5**:2
specifying general model, **5**:31
U-CARE, **5**:17

— *I* —

individual covariate
SE, **B**:17
individual covariates, **11**:1
design matrices
column functions, **11**:20
design matrix, **11**:14
beer consumption, **11**:35
extensions, **11**:18
introduction, **11**:7
design matrix functions, **11**:23
deviance plots, **11**:46
goodness of fit, **11**:46
linear models
classification factors, **11**:36
missing values, **11**:29, **11**:30
model averaging, **11**:44
normailizing selection example, **11**:14
normalizing selection example, **11**:3
plotting, **11**:23
reconstituting SE
Delta method, **B**:15, **B**:20
standard errors
Delta method, **11**:14
standardizing, **11**:9
problems, **11**:10
time-varying covariates, **11**:29
multi-state approach, **11**:30
normalizing selection example, **11**:30
input files
editors, **2**:2
generating, **2**:10

— *J* —

Jolly-Seber models, **12**:1
future directions, **12**:50
goodness-of-fit, **12**:18
Introduction, **12**:1
Link-Barker and Pradel recruitment, **12**:7
model types, **12**:3
Burnham and Pradel-lambda, **12**:9
classical Jolly-Seber, **12**:4
POPAN, **12**:5
summary of differences, **12**:11
multinomial link function, **12**:24
POPAN
deviance, **12**:13
POPAN vs Link-Barker, **12**:29
Pradel estimates versus other models, **12**:35
sampling protocol, **12**:1
Jolly-Sever models
POPAN model
multinomial link function, **12**:24

— *K* —

known-fate data, **16**:1
binomial model, **16**:3
Burnham models, **16**:23
censoring, **16**:25
derived parameters, **16**:27
encounter history, **16**:5
goodness of fit, **16**:26
Kaplan-Meier, **16**:1
nest success, **16**:27
radio impact, **16**:25
staggered entry, **16**:11
design matrix, **16**:19
worked example, **16**:11
temporary emigration, **16**:25

— *L* —

likelihood ratio test, **1**:20
linear models
> 1 classification factor, **6**:54
β estimates
odds-ratios, **6**:69
additive models, **6**:58
age models, **7**:14
constraint
definition, **6**:1
design matrix
degrees of freedom, **6**:17
design matrix vs PIMs, **6**:38
full vs. reduced, **6**:15
parameter estimability, **6**:77
subset models, **6**:33
effect size, **6**:61
AIC vs P-values, **6**:70
effect of $\hat{c}$, **6**:70
graphing, **6**:72
standard error, **6**:65
individual covariates
classification factors, **11**:36
introduction, **6**:1-7
linear regression, **6**:1
link functions

cumulative logit, **6**:47
definition, **6**:8, **6**:11
design matrix, **6**:22, **6**:23
matrix approach, **6**:4
mean parameter value, **6**:80
bias, **6**:81
modeling approaches, **6**:82
nestedness, **6**:42
odds-ratios, **6**:69
real covariates, **6**:40
linear trend, **6**:44
reconstituting parameters, **6**:25, **6**:26
95% CI, **6**:29
Delta method, **6**:27
regression
ANOVA, **6**:8
null hypothesis, **6**:3
RMark, **6**:85
sequential approach to construction, **6**:71
trend
cumulative logit link, **6**:47
link function
multinomial, **12**:24
link functions, *see* linear models, *see* linear models
back-transforms, **6**:22
cumulative logit link, **6**:47
design matrix, *see* linear models, **6**:23
sin link, **6**:23
introduction, **6**:11-12
multinomial, **10**:23
options, **6**:22
Pradel models, **13**:9
LRT
derivation, **4**:51

— ***M*** —

mark-resight models
alternative to NOREMARK, **18**:2
no individual marks, **18**:5
overview, **18**:1
robust-design, **18**:1
suggestions, **18**:34
which one to use?, **18**:34
maximum likelihood
binomial, **1**:6-12
closed and non-closed form, **1**:9
Hessian, **1**:19
individual covariates, **11**:2
least-squares
AIC, **4**:34
mark-recapture example, **1**:9, **1**:15
multinomial, **1**:13-14
statistical tests, **1**:20
variance
sample size, **1**:11
variance estimation
closed-form solutions, **1**:9
multiple parameter models, **1**:19
single parameter models, **1**:12, **1**:20
variance-covariance matrix, **1**:19
MCMC
caveats, warnings, and recommendations, **E**:46
ML estimates, **E**:15
multivariate hyperdistributions, **E**:35
autocorrelation analysis, **E**:39
back-transformation of correlation, **E**:45
back-transforming correlation, **E**:42
dead recovery example, **E**:36
Pradel model example, **E**:43
variance components analysis, **E**:5
>1 hyperdistribution, **E**:26
back-transforming estimates of mean and sigma, **E**:19
estimating mean and sigma, **E**:19
MCMC in MARK, **E**:1
model averaging
closed population models, **14**:27
confidence intervals, **14**:35
confounded parameters, **4**:45
Delta method, **B**:27
introduction, **4**:40
model selection uncertainty, **4**:47
non-interactive, **4**:47
random effects models, **D**:40
unconditional SE, **4**:47
variance components, **4**:47
model deviance
goodness of fit, **5**:4
model selection
$\hat{c}$
rules-of-thumb, **5**:37
AIC
conceptual basis, **4**:30
evidence ratios, **4**:39
introduction, **4**:30
least-squares, **4**:34
model weights, **4**:37
precision-fit tradeoff, **4**:30
quasi-likelihood, **5**:5
refinements, **4**:33
rules-of-thumb, **4**:39
AIC versus BIC
example, **11**:28
AIC versus LRT
overview, **4**:59
BIC, **4**:35

likelihood-ratio test, **4**:51
nested models, **4**:54
model averaging, **4**:40
nested models, **4**:54
LRT, **4**:51
precision-fit tradeoff, **4**:29
significance testing, **4**:50
problems..., **4**:55
multi-state models
basic assumptions, **10**:4, **10**:5
CJS as MS problem, **10**:28
cost of breeding example, **10**:6-10
dead-recovery as MS problem, **10**:31
design matrix
trend among states, **11**:32
design-matrix, **10**:12
goodness of fit
Arnason-Schwarz model, **10**:46
U-CARE, **10**:47
memory models, **10**:12
metapopulation example, **10**:17-23
multinomial link function, **10**:23
numerical estimation
local minima, **10**:39
MCMC, **10**:41
simulated annealing, **10**:39
omnibus framework, **10**:27
recruitment example, **10**:35
specifying the transitions, **10**:11
unequal intervals, **10**:48
unobservable state example, **10**:35
multivariate
reconstituting SE
Delta method, **B**:17

— ***N*** —

nest survival
design matrix
column functions, **17**:18
effective sample size, **17**:8
encounter history, **17**:4
goodness of fit, **17**:19
Mayfield method, **17**:1-3
observor effects, **17**:16
telemetry, **17**:18
nested models
definition, **4**:54
numerical estimation
simulated annealing, **10**:39
starting values, **10**:39

— ***P*** —

parameter counting
parameter estimability, *see* design matrix
parameter estimability
closed population models, **14**:40
Parameter index matrices, **4**:2-10
PIM chart, **4**:12
parameter indexing
parameter index matrix, *see* PIMs
PIMs, **3**:10
power analysis
simulations, **A**:10
practice data sets, ii
Pradel models
CLogit link, **13**:9
cautionary notes, **13**:16
deriving λ, **13**:5
link functions, **13**:9
partitioning contributions, **13**:14
population λ vs. Pradel's λ, **13**:6
population growth rates
definitions, **13**:1
estimation from CMR data, **13**:2
reverse encounter histories, **13**:3, **13**:4
pre-defined models, **3**:13
Program MARK
discussion forum, iv
documentation, iv, v
background reading, v
installation, iii
upgrading, iii

— ***R*** —

Random effects models, **D**:1
random effects models
caveats, warnings, and recommendations, **D**:42
AIC, **D**:23
binomial survival
simple mean, **D**:7
simple trend, **D**:12
known-fate example, **D**:7, **D**:12
model averaging, **D**:40
variance components
background, **D**:3
random variables
moments, **B**:1
return rates, **1**:2-3
RMark
linear models, **6**:85
robust design
γ parameters
multi-state notation, **15**:7
'even flow' model, **15**:11
classical, **15**:4
identifiability constraints, **15**:12

summary, **15**:12
no movement model, **15**:11
advantages of, **15**:12
assumptions, **15**:13
design matrix, **15**:20
estimating γ, **15**:4
even flow model, **15**:35
goodness of fit, **15**:47
introduction, **15**:1
mark-resight models, **18**:1
Markovian models, **15**:6, **15**:9
probability expressions, **15**:9
motivation, **15**:1
multi-state closed design, **15**:29
open multi-state, **15**:39
open-MS
unobservable states, **15**:45
specifying primary and secondary samples, **15**:14
temporary emigration, **15**:6
time intervals, **15**:14
unequal time intervals, **15**:49
robust design:open-MS
parameter estimation, **15**:46

— *S* —

saturated models, **5**:2
Score test, **1**:20
shrinkage estimates
95% confidence limits, **D**:21
simple introduction to, **D**:19
technical background, **D**:21
significance testing, **4**:50
AIC, **4**:57
model selection
likelihood-ratio test, **4**:51
problems..., **4**:55
simulation
data bootstrap, **G**:1
simulations
closed-capture, **A**:18
generating encounter histories, **A**:12, **A**:17
power analysis, **A**:10
Program MARK, **A**:1-14
Program RELEASE, **A**:10
robust design, **A**:18

— *T* —

time intervals
data formatting, **2**:5
modeling, **4**:25

— *V* —

variance components
caveats, warnings, and recommendations, **D**:42
background, **D**:3
binomial survival
simple trend, **D**:12
known-fate example, **D**:12
variance estimation
profile likelihood, **1**:22, **F**:5

— *W* —

Wald test, **1**:20

— *Y* —

young survival models
assumptions, **19**:2
data formatting, **19**:2
motivation, **19**:1
parameters and sample size, **19**:5