Maxime HERVÉ

Aide-mémoire de statistique appliquée à la biologie

*Construire son étude et analyser les résultats à l'aide du logiciel **R***

Version finale (2016)
mise à jour 13.12.2021

AVANT-PROPOS

Il n'est pas besoin de grandes enquêtes d'opinions pour se rendre compte que les biologistes sont globalement frileux à se frotter aux statistiques. L'étape de l'analyse des résultats est souvent vécue comme une contrainte, un passage obligé mais désagréable, voire même parfois un calvaire. Pourtant, le premier objectif des statistiques est bien de révéler ce que les données ont a nous dire. Passer à côté d'une bonne analyse par manque de temps, de motivation ou de compétence, c'est surtout prendre le risque de rater un phénomène intéressant qui était pourtant là, sous nos yeux.

L'objectif de cet aide-mémoire est de guider tout biologiste qui en sentirait le besoin dans sa démarche statistique, depuis la construction du protocole expérimental jusqu'à l'analyse des résultats qui en découlent. Il doit permettre de s'en sortir seul, tout en assurant une analyse appropriée et rigoureuse. Bien entendu, il ne dispense pas de se poser des questions et il est toujours nécessaire d'adapter un minimum le code proposé à ses propres données. Pour imager les choses, considérez que vous apprenez à faire du vélo et que ce document est la paire de roulettes qui vous évite de chuter. C'est rassurant, mais n'oubliez pas qu'avant tout c'est vous qui pédalez.

Depuis la rédaction de la première version de cet aide-mémoire en 2010, j'ai beaucoup enseigné, formé et conseillé en statistique appliquée à la biologie. Ces nombreuses interactions avec des collègues et étudiants m'ont conduit à la structure de cette sixième et dernière version, qui je crois est celle qui est la plus adaptée à une bonne démarche d'analyse. Elle ne sera désormais modifiée qu'à la marge (les nouveautés de la présente version sont listées page suivante).

Un assez grand nombre de méthodes sont couvertes par ce document, et j'ai tout fait pour simplifier l'orientation dans cette « jungle ». Malgré tout, une bonne part du cheminement est dépendante du biologiste lui-même. Une bonne analyse est une analyse qui répond pertinemment à une question précise. La règle d'or est donc avant tout de bien identifier cette question ou série de questions, et de ne jamais l'oublier car le choix de la méthode statistique la plus pertinente en dépend et il est facile de se perdre en analysant ses données.

Cet aide-mémoire est directement associé au package `RVAideMemoire` *(voir fiche **8** pour son installation). La présente version du document correspond aux versions ≥ 0.9-81 du package.*
*Remarque importante : cette version de l'aide-mémoire a été actualisée après la sortie de la version 4 de **R** (2020). Je vous conseille fortement de mettre à jour **R** et d'en installer la version 4 (voir fiche **10**), qui introduit quelques changements majeurs.*

J'espère sincèrement que ce document comblera vos attentes et qu'il vous permettra de vous sentir moins seul dans le monde pas si cauchemardesque des statistiques. Pour commencer votre exploration, rendez-vous page 4.

Le 13 décembre 2021

Maxime HERVÉ

NOUVEAUTÉS DE LA PRÉSENTE VERSION DE L'AIDE-MÉMOIRE

— Ajout d'une page « Nouveautés de la présente version de l'aide-mémoire ».
— Ajout d'une page « Comment utiliser cet aide-mémoire ».
— Ajout d'une note sur l'installation du package `RVAideMemoire` (voir fiche **8**).
— Ajout de l'argument `stringsAsFactors=TRUE` à la création ou l'importation d'un tableau contenant des facteurs (voir fiches **5** et **6**).
— Ajout de quelques subtilités dans le calcul de moyennes ajustées basées sur un modèle (voir fiche **43**).
— Ajout de la possibilité de calculer un QAIC(c) pour les GLMs basés sur une loi quasi-binomiale ou quasi-Poisson (voir fiche **44**).
— Ajout de la fonction de lien logit empirique pour les GLMs basés sur une loi binomiale, pour les cas de séparation (quasi-)complète entre modalités d'un facteur (voir fiches **48** et **60**).
— Réécriture de la fiche sur l'analyse d'une réponse continue bornée. L'analyse est désormais basée sur une méthode réellement dédiée à ce type de réponse, la régression bêta (voire fiche **64**).

COMMENT UTILISER CET AIDE-MÉMOIRE

J'ai tout fait pour faciliter le parcours de ce document assez volumineux. Il est ainsi bourré de liens, à côté desquels passent de nombreux lecteurs et c'est bien dommage ! Ainsi :

— Tout ce qui apparaît **en gras et rouge** est un lien. Cela concerne *tous les numéros de fiches* (ex : fiche **1**), ainsi que les parties du sommaire (page suivante). Les fiches renvoient ainsi directement les unes vers les autres, sans besoin de faire défiler les pages du document.

— En haut à gauche de *tous les sommaires de parties*, ainsi que de *toutes les fiches*, un symbole  renvoie au sommaire général.

— En haut à gauche de *toutes les fiches*, un lien renvoie vers le sommaire de la partie concernée.

Pour ce qui est du fond, le plus important est de se laisser guider par les sommaires successifs. Il est malheureux que ces sommaires soient souvent survolés pour aller rapidement – parfois trop instinctivement – vers les fiches. Si vous ne savez pas déjà parfaitement où aller avant même d'ouvrir ce document, prenez le temps de lire ces sommaires.

Dans la pratique, la plupart des lecteurs sont intéressés par la Partie III « Analyser les résultats d'une étude ». Le sommaire de cette partie explique clairement comment choisir entre analyses univariées, bivariées et multivariées. Les mondes univariés et multivariés sont particulièrement riches. Dans le sommaire de ces deux sous-parties, des encadrés sont là pour vous guider vers les fiches les plus pertinentes selon la question à laquelle l'analyse statistique doit répondre. Ces encadrés constituent *l'information la plus cruciale de tout l'aide-mémoire*, et à ce titre ils doivent être lus avec attention.

SOMMAIRE

PARTIE I – BASES DU FONCTIONNEMENT DE R

Ce document n'est pas à proprement parler une introduction à R. Cette partie rappelle seulement quelques notions essentielles comme la manipulation des objets courants (vecteurs, tableaux, matrices et listes), la construction et l'importation d'un jeu de données, la gestion des packages et diverses autres choses comme des « bonnes pratiques ».

PARTIE II – THÉORIE STATISTIQUE ÉLÉMENTAIRE

Ce document n'est pas non plus une introduction aux statistiques. Cependant certaines bases théoriques sont indispensables pour construire une étude proprement et en analyser correctement les résultats : types de variable, plan d'échantillonnage ou d'expérience, fonctionnement d'un test, taille de l'échantillon à constituer. Accessoirement, les lois de distribution les plus courantes sont présentées.

PARTIE III – ANALYSER LES RÉSULTATS D'UNE ÉTUDE

L'essentiel du document est dans cette partie, qui détaille comment représenter, synthétiser et analyser des données à une, deux ou plus de deux dimensions.

Première partie

Bases du fonctionnement de R

MANIPULATION DES OBJETS COURANTS

PRÉPARATION ET IMPORTATION DES DONNÉES

DIVERS

1. Les vecteurs

Le vecteur est à la fois l'objet le plus simple et le plus fondamental du langage **R**. Il se crée grâce à la fonction `c()`, qui prend comme arguments les éléments du vecteur. Tous ces éléments doivent être *du même type* : valeurs numériques, chaînes de caractères ou encore niveaux d'un facteur.

EXEMPLE(S)

Pour créer un vecteur numérique :

```
> vecteur <- c(7,9,4,12,18)
> vecteur
[1] 7 9 4 12 18
```

Pour créer un vecteur de chaînes de caractères :

```
> vecteur <- c("H","C","I","G","F")
> vecteur
[1] "H" "C" "I" "G" "F"
```

Pour créer un facteur :

```
> vecteur <- factor(c("niv1","niv2","niv2","niv3","niv1"))
> vecteur
[1] niv1 niv2 niv2 niv3 niv1
Levels: niv1 niv2 niv3
```

Il existe des fonctions ou des abréviations qui permettent de simplifier la création de certains vecteurs usuels :

EXEMPLE(S)

```
> 1:10
[1] 1 2 3 4 5 6 7 8 9 10
> seq(from=1,to=3,by=0.25)
[1] 1.00 1.25 1.50 1.75 2.00 2.25 2.50 2.75 3.00
> LETTERS[1:5]
[1] "A" "B" "C" "D" "E"
```

Pour accéder au(x) $i^{ème(s)}$ élément(s) d'un vecteur, écrire `vecteur[i]`, où `i` peut être une valeur unique ou lui-même un vecteur :

EXEMPLE(S)

```
> vecteur <- seq(from=2,to=16,by=2)
> vecteur
[1] 2 4 6 8 10 12 14 16
> vecteur[5]
[1] 10
> vecteur[c(2,5,8)]
[1] 4 10 16
> vecteur[-c(2,5,8)]
[1] 2 6 8 12 14
> vecteur[6:3]
[1] 12 10 8 6
```

2. Les tableaux

Les tableaux sont simplement un moyen de regrouper (en colonnes) des vecteurs dans le même objet, chaque colonne étant *indépendante*. L'unique contrainte est que tous les vecteurs doivent avoir *la même longueur*.

Pour créer un tableau, utiliser la fonction `data.frame()`, qui prend en arguments les différentes colonnes (de gauche à droite). On peut préciser le titre des colonnes. Dans le cas d'un vecteur de chaînes de caractères, celui-ci n'est transformé en facteur que lorsque l'argument `stringsAsFactors=TRUE` est ajouté. Puisque cette transformation est nécessaire dans l'immense majorité des cas, mieux vaut prendre l'habitude de toujours utiliser cet argument.

Remarque : avant la version 4.0.0 de **R**, l'argument `stringsAsFactors` avait pour valeur par défaut `TRUE`. Ceci a changé à partir de la version 4.0.0, ce qui explique la nécessité de le préciser désormais.

EXEMPLE(S)

```
> variable1 <- 1:5
> variable2 <- LETTERS[1:5]
> tableau <- data.frame(variable1,variable2,stringsAsFactors=TRUE)
> tableau
  variable1 variable2
1         1         A
2         2         B
3         3         C
4         4         D
5         5         E
```

Le tableau peut être créé directement *via* :

```
> tableau <- data.frame(variable1=1:5,variable2=LETTERS[1:5],stringsAsFactors=
TRUE)
```

Pour accéder à un (ou plusieurs) élément(s) d'un tableau, le principe est le même que pour les vecteurs (voir fiche **1**) excepte qu'il n'y a pas une mais *deux* dimensions à l'objet (*i.e.* les lignes et les colonnes). Le principe d'indexation est valable pour tous les objets à deux dimensions et est celui-ci : `tableau[ligne(s),colonne(s)]`, où `ligne(s)` et `colonne(s)` sont soit des valeurs uniques, soit des vecteurs. Si rien n'est mis avant la virgule toutes les lignes sont sélectionnées, si rien n'est mis après toutes les colonnes sont sélectionnées.

EXEMPLE(S)

```
> tableau[c(1,3),]
  variable1 variable2
1         1         A
3         3         C

> tableau[c(3,5),2]
[1] C E
Levels: A B C D E
```

Dans le cas particulier de la sélection d'une colonne entière, il y a trois autres possibilités :

— `tableau$colonne` où `colonne` est le *nom* de la colonne
— `tableau$"colonne"` où `colonne` est le *nom* de la colonne, entre guillemets
— `tableau[,"colonne"]` où `colonne` est le *nom* de la colonne, entre guillemets.

3. Les matrices

À la différence des tableaux (voir fiche **2**), les matrices sont un tout cohérent, *i.e.* les colonnes ne sont pas indépendantes. Cela implique que *tous les éléments d'une matrice sont de même type* : numérique, texte, niveaux de facteur…

Pour créer une matrice, utiliser la fonction `matrix()`, qui prend comme arguments obligatoires les valeurs qui doivent la remplir, et le nombre de lignes et/ou de colonnes. Par défaut les matrices sont remplies en colonnes, pour les remplir en lignes ajouter l'argument `byrow=TRUE`. Pour donner un nom aux lignes et aux colonnes, utiliser l'argument `dimnames=list(lignes,colonnes)`, où `lignes` et `colonnes` sont des vecteurs :

EXEMPLE(S)

```
> matrice <- matrix(1:8,nrow=2)
> matrice
     [,1] [,2] [,3] [,4]
[1,]    1    3    5    7
[2,]    2    4    6    8
> matrice <- matrix(1:8,nrow=2,byrow=TRUE)
> matrice
     [,1] [,2] [,3] [,4]
[1,]    1    2    3    4
[2,]    5    6    7    8
> matrice <- matrix(1:8,nrow=2,dimnames=list(letters[1:2],LETTERS[1:4]))
> matrice
  A B C D
a 1 3 5 7
b 2 4 6 8
```

Il est également possible de créer des matrices à partir de plusieurs vecteurs qui doivent en constituer les lignes ou les colonnes. Utiliser pour cela les fonctions `rbind()` ou `cbind()`, qui assemblent les vecteurs respectivement en lignes et en colonnes :

EXEMPLE(S)

```
> vecteur1 <- 1:3
> vecteur2 <- 4:6
> matrice <- rbind(vecteur1,vecteur2)
> matrice
         [,1] [,2] [,3]
vecteur1    1    2    3
vecteur2    4    5    6
> matrice <- cbind(vecteur1,vecteur2)
> matrice
     vecteur1 vecteur2
[1,]        1        4
[2,]        2        5
[3,]        3        6
```

Les matrices étant des objets à *deux* dimensions (les lignes et les colonnes), leur indexation est identique à celle des tableaux (voir fiche **2**).

4. Les listes

Les listes sont des objets à compartiments, où chaque compartiment est totalement *indépendant* des autres. Une liste peut donc à la fois contenir un vecteur dans un compartiment, un tableau dans un autre, et même une liste dans un troisième.

Pour créer une liste, utiliser la fonction `list()`, qui prend en arguments ce que l'on veut mettre dans chaque compartiment (du premier au dernier). On peut préciser un nom à chaque compartiment, ce qui aide grandement à s'y retrouver.

EXEMPLE(S)

```
> vecteur <- 1:5
> tableau <- data.frame(v1=1:3,v2=LETTERS[1:3],stringsAsFactors=TRUE)
> list(vecteur,tableau)
[[1]]
[1] 1 2 3 4 5

[[2]]
 v1 v2
1 1  A
2 2  B
3 3  C
```

La liste peut être créée directement *via* :

```
> list(1:5,data.frame(v1=1:3,v2=LETTERS[1:3],stringsAsFactors=TRUE))
```

Pour donner un nom aux compartiments :

```
> liste <- list(A=1:5,B=data.frame(v1=1:3,v2=LETTERS[1:3],stringsAsFactors=TRUE))
> liste
$A
[1] 1 2 3 4 5

$B
 v1 v2
1 1  A
2 2  B
3 3  C
```

Pour accéder à un compartiment d'une liste (toujours un seul à la fois), on peut utiliser le numéro du compartiment entre *doubles* crochets : `liste[[i]]` où `i` est une valeur numérique unique. Si les compartiments de la liste ont un nom, on peut également utiliser la syntaxe `liste$nom` où `nom` est le nom du compartiment.

EXEMPLE(S)

```
> liste[[1]]
[1] 1 2 3 4 5
```

Ou, puisque les compartiments de cette liste ont un nom :

```
> liste$A
[1] 1 2 3 4 5
```

Pour accéder à un (ou des) élément(s) particulier(s) d'un compartiment d'une liste, il suffit de cumuler l'indexation de la liste et l'indexation de l'objet contenu dans le compartiment en question.

EXEMPLE(S)

```
> liste[[1]][c(2,4)]
[1] 2 4
> liste$B$v1[3]
[1] 3
```

5. La construction d'un tableau de données

La construction d'un tableau de données correctement structuré est une étape importante de l'étude, car si elle est mal réalisée elle peut mener à des résultats faux, ou le plus souvent à des erreurs une fois dans **R**.

Cette construction nécessite de se poser une question essentielle : quelles sont les variables prises en compte dans l'étude ? Y répondre implique d'identifier les variables quantitatives et les facteurs, ainsi que les classes des facteurs. Si les choses sont claires, l'analyse statistique le sera également.

D'une manière générale, il est conseillé de toujours construire son tableau de données dans un tableur. Cela permet d'enregistrer le jeu de données dans un fichier externe à **R**, et donc de toujours pouvoir y revenir puisque **R** ne modifie pas les fichiers externes (sauf si on le lui demande explicitement).

Dans le tableur, la règle est simple : les individus doivent être placés en *lignes* et les variables en *colonnes*.

Il est conseillé de donner un titre à chaque colonne, qui deviendra le nom de la variable dans **R**. Il est indispensable cependant de respecter certaines règles : les noms de variable ne doivent contenir ni espace, ni caractère accentué, ni symbole (ceci est une règle pour tous les noms d'objet dans **R**). Si un nom de variable doit contenir deux mots, ils peuvent être séparés par un point (.) ou un tiret bas (_). Mieux vaut également privilégier les noms courts mais clairs, car une fois dans **R** taper sans cesse des noms de variable longs est vite fastidieux.

Le tableau de données doit absolument obéir à une autre règle : *aucune case ne doit être vide*. La seule exception possible est celle en haut à gauche si les colonnes ont un titre, auquel cas la 1ère colonne sera comprise par **R** comme le nom des lignes. S'il manque une donnée pour un individu, il faut se demander d'où elle vient :

— si c'est une donnée inutilisable (mesure ratée, mal retranscrite...), pas de problème. On dit alors qu'on a une « donnée manquante », que l'on doit noter `NA` (pour *Not Available*, *i.e.* donnée manquante). Le tableur comme **R** reconnaissent le `NA`, qu'ils interprètent correctement.

— si la situation est autre, c'est que le tableau est mal construit et qu'en particulier les variables n'ont pas été bien définies. La réflexion s'impose donc pour identifier les variables et reconstruire un tableau de données.

Il est déconseillé de coder les niveaux d'un facteur avec uniquement des chiffres. **R** comprendrait cette variable comme numérique (et non comme un facteur), ce qui pourrait sérieusement perturber voire empêcher l'analyse.

Si des analyses dans **R** doivent se faire uniquement sur un sous-ensemble du tableau de données, ou si pour certaines analyses le tableau de données serait plus facile à utiliser s'il était construit autrement, il est conseillé de construire plusieurs tableaux de données séparés. Il est toujours possible de manipuler le tableau initial dans **R** pour en extraire une partie ou pour le transformer, mais il est clairement plus facile (et surtout moins source d'erreur) lorsque l'on n'a pas l'habitude de le faire en amont, dans le tableur.

6. L'importation d'un tableau de données dans R

Il existe de nombreuses méthodes pour importer ses données dans R. Une seule est présentée ici, qui est à la fois très simple, fonctionne dans la plupart des situations et peut être utilisée sur toutes les plateformes.

La procédure se fait en trois étapes :

1. Dans le tableur, sélectionner toutes les cases constituant le tableau de données.
2. Copier ce tableau dans le bloc-notes et enregistrer le fichier en format `.txt`.
3. Dans R, charger le tableau de données grâce à la fonction `read.table()` et le stocker dans un objet : `tableau<-read.table("fichier")` où `fichier` est le nom du fichier texte (avec l'extension `.txt` et éventuellement le chemin qui mène à ce fichier), entre guillemets.

R étant un logiciel anglo-saxon, le séparateur décimal qu'il utilise est le point. Or dans les tableurs français (et donc dans le fichier texte) le séparateur décimal est la virgule. Si le tableau de données contient des valeurs décimales, il est donc nécessaire de préciser à R qu'il interprète la virgule comme séparateur décimal. Ajouter pour cela l'argument `dec=","` à la fonction `read.table()`.

Si les colonnes du tableau de données ont un titre, qui doit donc être interprété comme le nom de la variable, ajouter l'argument `header=TRUE`.

Si le tableau contient des colonnes de chaînes de caractères qui doivent être interprétées comme des facteurs, ajouter l'argument `stringsAsFactors=TRUE`.

Puisque dans l'immense majorité des cas, un tableau contient des valeurs décimales, ses colonnes ont un titre et certaines variables sont des facteurs, il est bon de prendre l'habitude de toujours utiliser les arguments `dec=","`, `header=TRUE` et `stringsAsFactors=TRUE`.

Remarque : avant la version 4.0.0 de R, l'argument `stringsAsFactors` avait pour valeur par défaut `TRUE`. Ceci a changé à partir de la version 4.0.0, ce qui explique la nécessité de le préciser désormais.

Une fois le tableau importé, il est indispensable de vérifier qu'il n'y a pas eu d'erreur pendant son chargement. Pour cela appeler le résumé du tableau *via* `summary(tableau)`. R renvoie un résumé de chaque variable :

— pour une variable numérique, R donne des indications sur sa distribution : minimum, 1er quartile, médiane, moyenne, 3ème quartile et maximum.
— pour un facteur, R donne le nombre d'individus par classe.

Si un facteur est codé numériquement (par exemple un facteur binaire ou un facteur ordinal), R l'interprète comme une variable numérique. Pour transformer cette variable en facteur, taper `tableau$variable<-factor(tableau$variable)` où `variable` est le nom de la variable.

7. Bonnes pratiques

Il existe des dizaines d'astuces ou de façon de procéder pour utiliser R au mieux. Certaines, importantes, sont résumées ici.

Logiciel et packages

Les packages ne sont pas gravés dans le marbre mais évoluent avec le temps. Les auteurs corrigent des erreurs, ajoutent de nouvelles fonctions... Pour bénéficier de ces améliorations/ajouts/corrections, il est nécessaire de mettre à jour régulièrement ses packages (une fois par mois est un bon rythme). Voir fiche **8** pour la procédure, très simple.

Le logiciel R lui-même évolue. Le numéro de la version installée est toujours donné dans le message d'accueil au lancement du logiciel, par exemple 3.3.1. Le premier chiffre de ce numéro est très important. En effet, toutes les versions des packages créées après la sortie d'une version V.x.x. de R ne sont *pas disponibles* pour les utilisateurs des versions [V-1].x.x. Ces mises à jour dites « majeures » du logiciel ont donc une importance réelle. Il est fortement recommandé de suivre ces mises à jours majeures, qui sont tout de même relativement rares. Certains packages exigent cependant une version minimale de R pour pouvoir être mis à jour. Installer une nouvelle version du logiciel régulièrement (une fois tous les 1 à 2 ans) permet d'avoir accès à toutes les mises à jour récentes (voir fiche **10**).

Création des objets

Au-delà des règles obligatoires sur les noms d'objet (ne pas commencer par un chiffre, pas d'espace...), quelques principes permettent d'éviter des erreurs qui peuvent parfois conduire à une analyse statistique biaisée (ou en tout cas à se compliquer la vie) :

- toujours donner un nom informatif aux objets, pour s'y retrouver plus facilement (attention tout de même aux noms à rallonge qui au final font perdre du temps !)
- toujours créer ses objects avec la syntaxe `nom <- contenu`, et pas `nom = contenu`
- ne jamais appeler deux objets par le même nom
- ne jamais donner à un objet le nom d'une fonction
- toujours coder les facteurs avec au moins une lettre (et pas sous forme numérique), pour que R reconnaisse bien ces variables comme des facteurs.

Utilisation des fonctions

Dès qu'une fonction accepte comme argument une *formule* (voir fiche **40** bien que les formules soient utilisées aussi dans d'autres fonctions que celles créant un modèle), elle a un argument `data`. Celui-ci permet de préciser le tableau de données dans lequel aller chercher les variables contennues dans la formule, ce qui simplifie la rédaction la formule (donc sa clarté) mais évite aussi de provoquer des erreurs.

EXEMPLE(S)

Au lieu de (peu importe le nom de la fonction) :

```
> lm(tableau$y~tableau$x+tableau$z)
```

Préférer la syntaxe :

```
> lm(y~x+z,data=tableau)
```

Quand une fonction accepte plusieurs arguments, mieux vaut les appeler explicitement par leur nom pour éviter d'attribuer à un argument une valeur qui était destinée à un autre argument. En effet, sans utiliser les noms d'arguments il faut *strictement* respecter l'ordre des arguments tels que précisés dans la page d'aide de la fonction, ce qui est souvent source d'erreurs. Le premier ou les deux premiers arguments échappent à cette règle car ils sont souvent tellement logiques que l'on risque peu de se tromper.

Divers

Éviter d'utiliser les fonctions `attach()` et `detach()`, qui sont source de bien des erreurs. Grâce à l'argument `data` des fonctions acceptant une formule, ou de certaines fonctions telles que `with()`, on peut se rendre la vie aussi simple sans risquer quoi que ce soit.

Enfin, il est très fortement recommandé d'enregistrer les scripts de ses analyses afin de pouvoir y revenir plus tard (ou au moins d'être sûr de ce que l'on a fait!). Il est une très bonne chose d'aérer ses scripts et de les commenter (tout ce qui est situé après le symbole **#** est reconnu par **R** comme un commentaire), pour gagner en clarté et en compréhensivité par d'autres personnes (ou soi-même plus tard). Ne pas oublier de préciser les packages nécessaires à l'analyse dans le script.

8. Installer, charger et mettre à jour des packages

Installer un package

Il est nécessaire d'être connecté à internet pour installer un package, car celui-ci doit être téléchargé depuis un serveur. L'installation ne se fait qu'*une seule fois*.

Si R est utilisé depuis la console R, taper `install.packages("package")` où `package` est le nom du package désiré, entre guillemets. Il est demandé ensuite de choisir un serveur (choisir « 0-Cloud [https] » est la meilleure assurance de télécharger la version la plus récente du package). Si le package nécessite que d'autres packages soient installés pour fonctionner (ces autres packages étant appelés « dépendances »), ces derniers seront installés automatiquement.

Si R est utilisé depuis la console système, la procédure se fait en deux étapes :

1. Télécharger les sources du package à partir de son site de dépôt, le site principal étant le CRAN (*the Comprehensive R Archive Network*) : *http://cran.r-project.org* rubrique *Packages*.
2. Installer le package en tapant `R CMD INSTALL package` où `package` est le nom du fichier `tar.gz` contenant les sources.

Attention, l'installation manuelle à partir des sources du package n'installent pas les dépendances de ce package !

La procédure expliquée ici est la plus simple, mais il existe de nombreuses variantes. Voir la R FAQ pour plus d'informations : *http://cran.r-project.org/doc/manuals/R-admin.html#Installing-packages*.

Petite subtilité à l'installation du package `RVAideMemoire`

Parmi les dépendances du package `RVAideMemoire` s'en trouve une qui n'est pas hébergé sur le CRAN : `mixOmics`. Pour installer `mixOmics`, taper les deux commandes suivantes :

```
> install.packages("BiocManager")
> BiocManager::install("mixOmics")
```

Une fois `mixOmics` installé, le package `RVAideMemoire` peut être installé :

```
> install.packages("RVAideMemoire")
```

Charger un package

Le chargement d'un package doit se faire *à chaque session où il doit être utilisé*. La commande est simple : `library(package)` où `package` est le nom du package, sans guillemets.

Mettre à jours les packages installés

Pour mettre à jour automatiquement tous les packages installés, taper `update.packages(ask=FALSE)`. R télécharge alors toutes les mises à jour et les installe. Attention, seuls les packages *non chargés* sont mis à jour. Mieux vaut donc faire les mises à jour avant toute autre chose lorsque R est ouvert.

Il est recommandé de mettre régulièrement à jour ses packages afin de profiter de leur évolution, souvent rapide.

9. Citer R et ses packages

Lors de l'écriture d'un document scientifique, il est une évidence de citer ses sources bibliographiques. Il doit également en être une de citer les logiciels utilisés lors de la réalisation de l'étude. **R** est certes gratuit, mais il n'en reste pas moins que des dizaines de personnes s'impliquent dans son développement, et qu'il est normal de faire honneur à leur travail en les citant.

R doit être cité dès lors qu'il est utilisé. Pour savoir comment le citer, il suffit de taper `citation()` et de recopier ce qui figure après `To cite R in publications use:`.

Concernant les packages, la règle implicite est de citer tous ceux qui ne sont pas chargés au démarrage de **R**. Cela comprend les packages installés avec **R** mais non chargés automatiquement, ainsi que ceux installés par l'utilisateur. Pour savoir comment les citer, taper `citation("package")` où `package` est le nom du package, entre guillemets. Recopier ce qui figure après `To cite the xxx package in publications use:`.

10. Changer de version de R

Pour installer une version plus récente de **R**, qui donne accès à des mises à jour de packages pas forcément disponibles avec une version plus ancienne (voir fiche **7**), la procédure se fait en plusieurs étapes :

1. Récupérer le chemin vers le dossier où sont installés les packages, *via* `.libPaths()` (dans ce dossier, il y a un sous-dossier par package).
2. Désinstaller **R**, comme n'importe quel autre logiciel. Tous les fichiers sont ainsi supprimés *à l'exception des packages installés*.
3. Installer la nouvelle version de **R**, téléchargeable depuis le site officiel : *https://cran.r-project.org*.
4. Ouvrir **R** et récupérer le chemin vers le dossier où sont installés les packages de cette nouvelle version, *via* `.libPaths()`. Ce chemin *n'est pas le même* que celui récupéré à l'étape 1, car il est spécifique à chaque version de **R**.
5. Copier tous les sous-dossiers (*i.e.* un par package) présents dans le dossier identifié à l'étape 1, et les coller dans le dossier identifié à l'étape 4. Ainsi tous les packages qui étaient installés sous la précédente version de **R** le sont désormais sous la nouvelle version.
6. Supprimer le dossier identifié l'étape 1, qui n'a plus d'utilité.
7. Mettre à jour tous les packages avant de faire quoi que ce soit d'autre (voir fiche **8**).

Deuxième partie

Théorie statistique élémentaire

NOTIONS GÉNÉRALES

11. Les différents types de variable
12. Le plan d'échantillonnage
13. Le plan d'expérience

FONCTIONNEMENT D'UN TEST STATISTIQUE

14. Principe des tests statistiques et risques associés à la conclusion
15. Le risque ou seuil de rejet α
16. La correction du seuil de rejet α (ou des *p-values*)
17. Risque β et puissance d'un test
18. Déterminer la taille de l'échantillon à constituer

LOIS DE PROBABILITÉS USUELLES

Lois de probabilité discontinues

19. Les lois de probabilité discontinues – généralités
20. La loi binomiale
21. La loi de Poisson
22. La loi binomiale négative

Lois de probabilité continues

23. Les lois de probabilité continues – généralités
24. La loi normale
25. La loi exponentielle
26. La loi de χ^2
27. La loi de Fisher - Snedecor
28. La loi de Student

11. Les différents types de variable

Il existe deux types de variable :

1. Quantitatives : leurs valeurs représentent une *grandeur*, quantifiable et le plus souvent associée à une unité de mesure. On peut effectuer des opérations mathématiques avec ces variables. Elles peuvent être de deux types :
   - continues, lorsqu'elles peuvent prendre une infinité de valeurs (dans un intervalle donné) : masse, temps, distance, volume…
   - discrètes, lorsqu'elles ne peuvent prendre que certaines valeurs (dans un intervalle donné) : nombre d'individus, d'évènements… Ces variables sont liées le plus souvent à des processus de comptage (où les valeurs prises ne peuvent être qu'entières et positives ou nulles).
2. Qualitatives : leurs valeurs ne représentent pas une quantité mais une *catégorie*. On ne peut donc pas effectuer d'opérations mathématiques avec ces variables. On les appelle des *facteurs*, et les valeurs qu'elles peuvent prendre des *classes*, *niveaux* ou *modalités*. Les variables qualitatives peuvent être de deux types :
   - ordinales, lorsque les classes peuvent être ordonnées : rang dans un classement, degré de satisfaction…
   - nominales, lorsque les classes ne peuvent pas être ordonnées : sexe, pays…

Les variables qualitatives peuvent être codées numériquement, mais il est très important de les différencier des variables quantitatives (car elles sont traitées différemment lors de l'analyse statistique). Un principe simple pour faire la différence : si l'on peut remplacer les valeurs d'une variable numérique par des énoncés, alors elle est qualitative. Par exemple, si l'on juge l'état d'une plante par une note allant de 0 à 5, on peut remplacer la note chiffrée par une appréciation du type « mauvais », « satisfaisant »… Cette variable est donc qualitative. Pour ne pas prendre de risque, il est conseillé de ne jamais coder les variables qualitatives numériquement.

Il existe deux types de facteur :

- à effet fixe (« facteur fixe ») : un facteur est fixe si ses classes ont été délibérément choisies, et si le but de l'étude est de les comparer. Par exemple, si l'on veut comparer la taille des individus entre trois espèces, le facteur « espèce » est fixe (à trois classes).
- à effet aléatoire (« facteur aléatoire ») : un facteur est aléatoire si ses classes ont été choisies parmi un grand nombre de classes possibles, et si le but de l'étude n'est pas de les comparer mais simplement de prendre en compte la variabilité qu'il existe entre elles. Par exemple, si les mesures de taille des trois espèces sont réalisées par quatre personnes différentes, on peut considérer un facteur « expérimentateur », aléatoire. L'objectif ici n'est en effet pas de comparer les mesures réalisées par les quatre personnes, mais de prendre en compte le fait que la façon de réaliser les mesures peut varier entre elles.

Il y a deux choses à bien garder à l'esprit : (i) la décision de déclarer un facteur comme fixe ou aléatoire est *fondamentale* pour l'analyse des données, car ce ne sont pas les mêmes analyses qui sont réalisées dans les deux cas ; (ii) cette décision doit être prise *selon l'objectif de l'étude, i.e.* la question à laquelle l'étude doit répondre, car aucun facteur n'est fixe ou aléatoire dans l'absolu. Il est donc indispensable de bien réfléchir avant de déclarer un facteur comme fixe ou aléatoire.

Remarque : dans le cas particulier d'un facteur à deux classes, on le déclare toujours comme fixe. Même si la logique de l'étude voudrait qu'il soit aléatoire, il n'est pas possible d'estimer de façon fiable l'effet d'un facteur aléatoire à si peu de classes. Trois classes est vraiment un minimum pour un facteur aléatoire, mais s'il est possible d'en avoir plus l'analyse y gagnerait en fiabilité.

Que ce soit pour des variables qualitatives ou quantitatives, si certaines mesures ne sont pas indépendantes entre elles, elles constituent des *séries appariées*. Le cas le plus simple est celui où plusieurs mesures sont réalisées sur un même individu (par exemple avant et après un traitement). Mais d'autres cas plus subtils peuvent se présenter, et il faut toujours réfléchir à la présence de séries appariées avant l'analyse. Il est en effet très important d'identifier les séries appariées lorsqu'elles existent, car des analyses statistiques adaptées doivent alors être utilisées. Les séries appariées sont le plus souvent identifiées par l'introduction d'un facteur aléatoire. Par exemple un facteur « individu » si plusieurs mesures sont réalisées par individu.

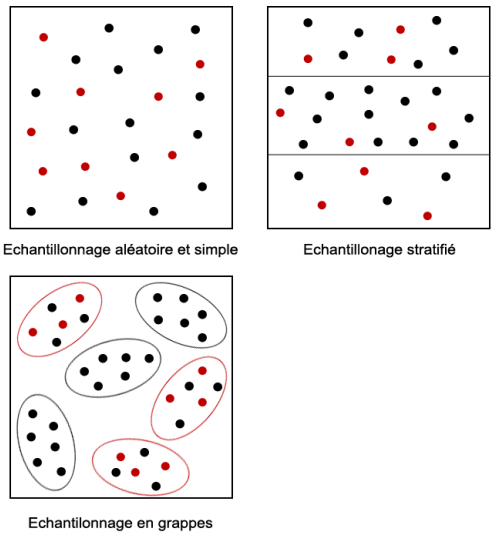
12. Le plan d'échantillonnage

On utilise un plan d'échantillonnage lorsque l'on réalise une étude par enquête, *i.e.* lorsque l'on collecte des informations sur un groupe d'individus dans leur milieu habituel, mais que tous les individus ne sont pas accessibles (par choix ou par contrainte).

Les principales méthodes d'échantillonnage peuvent être regroupées en deux ensembles :

1. L'échantillonnage aléatoire : tous les individus (au sens statistique) ont la même probabilité d'être choisis, et le choix de l'un n'influence pas celui des autres. Différentes méthodes d'échantillonnage aléatoire existent (voir les illustrations) :
   - l'échantillonnage aléatoire et simple : le choix se fait parmi tous les individus de la population (au sens statistique), qui ne forme qu'un grand ensemble.
   - l'échantillonnage stratifié : si la population est très hétérogène, elle peut être divisée en sous-ensembles exclusifs (ou strates). Au sein de ces strates l'échantillonnage est ensuite aléatoire et simple. Les strates sont identifiées dans l'analyse statistique comme les niveaux d'un facteur fixe.
   - l'échantillonnage en grappes : si les strates sont très nombreuses, on en choisit certaines au hasard (les grappes). Au sein de ces grappes l'échantillonnage est ensuite aléatoire et simple. Les grappes sont identifiées dans l'analyse statistique comme les niveaux d'un facteur aléatoire.
   - l'échantillonnage par degrés : il est une généralisation de l'échantillonnage en grappes (qui est en fait un échantillonnage du premier degré). Au sein de la population on choisit des grappes « primaires », puis à l'intérieur de celles-ci des grappes « secondaires » (toujours au hasard), et ainsi du suite... Au dernier niveau l'échantillonnage est aléatoire et simple.
2. L'échantillonnage systématique : un premier individu est choisi aléatoirement, puis les autres sont choisis de façon régulière à partir du précédent (dans le temps ou l'espace). L'analyse de ce type d'échantillonnage, qui fait appel à la statistique spatiale ou à l'analyse des séries chronologiques, n'est pas abordée dans cet ouvrage.

Il est important d'identifier la méthode mise en œuvre car les analyses statistiques doivent être adaptées. Seule l'analyse de plans d'échantillonnage aléatoires est abordée dans cet ouvrage.



Chaque point représente un individu. Les individus échantillonnés sont représentés en rouge.

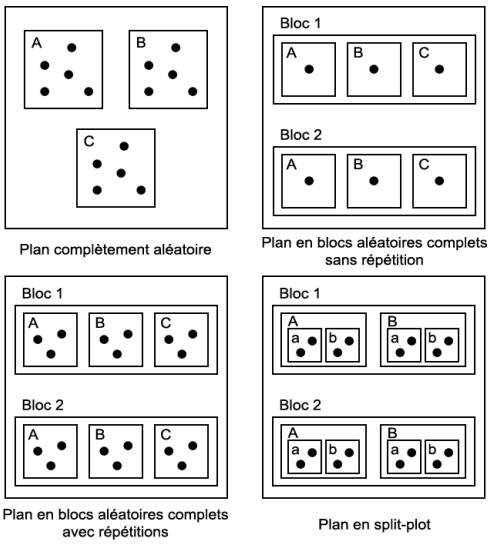
13. Le plan d'expérience

On utilise un plan d'expérience lorsque l'on réaliste une étude par expérimentation, *i.e.* lorsque l'on provoque volontairement les faits à étudier. Le plan d'expérience comprend notamment le(s) facteur(s) à faire varier, le nombre de répétitions à réaliser et le dispositif expérimental à mettre en place. L'association des classes de plusieurs facteurs constitue un *traitement*.

Il existe de nombreux types de dispositif expérimental, dont les principaux sont (voir les illustrations) :

— le plan d'expérience complètement aléatoire : chaque individu (au sens statistique) est affecté à un traitement aléatoirement.
— le plan d'expérience en blocs aléatoires complets : s'il y a (ou s'il peut y avoir) une grande hétérogénéité entre les individus, ils sont réunis en groupes aussi homogènes que possibles (ou blocs). Au sein de ces blocs chaque individu est ensuite affecté aléatoirement à un traitement, de manière à ce que tous les traitements soient présents dans chacun des blocs. Les blocs sont identifiés dans l'analyse statistique comme les niveaux d'un facteur aléatoire.
— le plan d'expérience en blocs aléatoires incomplets : dans ce cas tous les traitements ne sont pas présents dans chacun des blocs.
— le plan d'expérience en split-plot : le principe du split-plot est le plus souvent associé à celui des blocs aléatoires complets. Dans ce cas, dans chacun des blocs sont créés autant de sous-blocs qu'il y a de classes au premier facteur étudié. À chacun de ces sous-blocs est associée une classe. Puis chaque sous-bloc est divisé en autant d'unités qu'il y a de classes au second facteur étudié. À chacun de ces « sous-sous-blocs » est associée une classe. Pour plus de deux facteurs, la situation est plus complexe.

Quelle que soit la méthode employée, elle doit être clairement définie car elle doit être prise en compte dans les analyses statistiques.



Chaque point représente un individu. Chaque carré représente un traitement. Les niveaux du 1er facteur sont notés en majuscule, les niveaux du 2nd facteur sont notés en minuscule.

14. Principe des tests statistiques et risques associés à la conclusion

Le principe de réalisation de tout test statistique est le suivant :

1. On pose une hypothèse nulle H_0, de type « rien à signaler » (ex : les moyennes μ_A et μ_B sont égales) ou « valeur ponctuelle » (ex : la moyenne $\mu = 10$, la proportion $p = 50$ %).
2. On pose une hypothèse H_1, de telle manière que H_0 et H_1 soient exclusives (ex : les moyennes μ_A et μ_B sont différentes, la moyenne $\mu \neq 10$).
3. On calcule la valeur de la Variable de Test (VT), d'après une formule qui dépend du test utilisé.
4. On utilise la valeur de la VT calculée pour déterminer une *p-value*, *i.e.* une probabilité d'obtenir la valeur mesurée (moyenne, pourcentage...) si H_0 est vraie.
5. On conclut sur les deux hypothèses posées grâce à cette *p-value* :
   - si la *p-value* est supérieure au seuil α fixé avant le test (5 % en général, voir fiche **15**), on ne rejete pas H_0.
   - si la *p-value* est inférieure au seuil α, on rejete H_0.

Conclure sur les deux hypothèses présente deux risques :

- le risque de 1ère espèce ou risque α, qui correspond au risque de rejeter de H_0 si celle-ci est réellement vraie. On fixe ce risque, le plus souvent à 5 % (voir fiche **15**).
- le risque de 2ème espèce ou risque β, qui correspond au risque de ne pas rejeter de H_0 si celle-ci est réellement fausse. On ne peut pas fixer ce risque (voir fiche **17**).

	Réalité (inconnue le plus souvent)	
Décision	H_0 vraie	H_0 fausse
H_0 non rejetée	Bonne décision	Erreur β
H_0 rejetée	Erreur α	Bonne décision

La probabilité associée au fait de rejeter H_0 si celle-ci est fausse (soit $1-\beta$) est appelée *puissance* du test (voir fiche **17**).

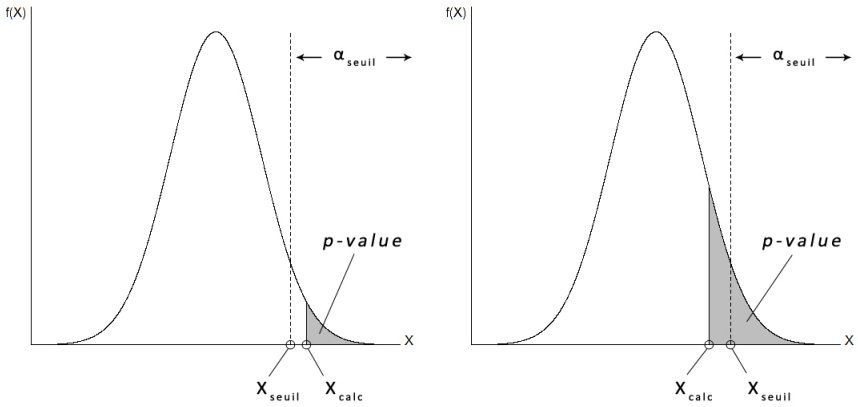
Il est important de distinguer les notions d'effet *statistiquement* significatif et *biologiquement* significatif. Statistiquement, on peut toujours rejeter H_0 moyennant d'avoir un échantillon suffisamment grand (voir fiche **17**). D'un point de vue statistique, H_0 n'est donc ni « vraie » ni « fausse ». Cette notion de vrai ou faux se pose uniquement en terme biologique. En pratique, on observe toujours un effet, aussi petit soit-il. La question est de savoir si cet effet (ou plutôt l'effet réel, qui n'est qu'estimé par l'expérimentation) est suffisamment grand pour qu'il ait une signification biologique. Si oui alors H_0 est réellement fausse, si non H_0 est réellement vraie.

15. Le risque ou seuil de rejet α

Le risque α, ou seuil de rejet ou encore seuil de signification de l'hypothèse H_0, est une valeur fixée arbitrairement *avant la réalisation de tout test statistique*. Elle correspond à un risque assumé de se tromper, celui de rejeter H_0 *si celle-ci est réellement vraie* (ce que bien sûr on ne sait jamais).

Lorsque l'on calcule une *p-value* suite à un test (voir fiche **14**), on la compare au seuil α choisi :
— si la *p-value* est inférieure au seuil α, on rejette H_0. Il faut être conscient que l'on prend un risque α de se tromper.
— si la *p-value* est supérieure au seuil α, on ne rejette pas H_0.

Obtenir une *p-value* supérieure au seuil α peut avoir deux origines :
— une réelle véracité de H_0 (au sens biologique, voir fiche **14**)
— un manque de puissance du test (voir fiche **17**), *i.e.* un risque (β, voir fiche **14**) important de « passer à côté » d'une réelle fausseté de H_0. L'effet observé peut donc être biologiquement significatif, mais la taille de l'échantillon insuffisante pour le montrer d'un point de vue statistique. D'où l'importance de déterminer la taille de l'échantillon *en fonction de ce seuil de significativité biologique*, dès que ce seuil est estimable (voir fiche **18**).



X_{seuil} : valeur de la Variable de Test (VT) X qui donne une fonction de répartition à droite égale au seuil α (test unilatéral droit ici).
X_{calc} : valeur de la VT X calculée à partir de l'échantillon testé.

À gauche l'hypothèse H_0 est rejetée, à droite elle ne l'est pas.

D'un point de vue pratique, on utilise souvent un seuil α de 5 %. Cela veut dire qu'on assume un risque de 5 % de rejeter H_0 si celle-ci est réellement vraie.

16. La correction du seuil de rejet α (ou des *p-values*)

Si une série de tests statistiques est réalisée avec à chaque fois α pour seuil de rejet de H_0 (voir fiche **15**), le risque global de rejeter H_0 si celle-ci est vraie augmente. En effet, plus on effectue de tests, plus on a de chance de tomber sur un échantillon peu représentatif de la population dont il provient (donnant une *p-value* inférieure au seuil α).

Il est donc nécessaire de corriger le seuil de rejet α de chaque test (ou leur *p-value*, ce qui revient au même) lorsque plusieurs sont réalisés, afin que le risque global soit égal au α souhaité.

Cette situation se présente :

— lorsque les tests vont permettre de prendre une décision unique, dès que l'un d'eux au moins permet le rejet de H_0

— lorsqu'est réalisée une série de tests deux-à-deux, soit directement soit après une analyse globale (ANOVA, test du χ^2 d'homogénéité...).

Plusieurs méthodes de correction existent, dont les trois suivantes.

La technique de Bonferroni

Si k tests sont effectués, la technique consiste simplement à diviser le seuil de rejet global α par k, donc considérer pour chaque test le seuil de rejet $\frac{\alpha}{k}$.

Cela revient à multiplier chaque *p-value* par k, sans changer le seuil α.

La technique séquentielle de Holm

La procédure se réalise en plusieurs étapes :

1. Classer les *p-values* de tous les tests réalisés par ordre croissant ($p_1 < \ldots < p_k$), k étant le nombre de tests effectués.
2. Rejeter H_0 pour les tests dont la *p-value* satisfait la condition :

$$p_i \leq \frac{\alpha_{\text{seuil}}}{k+1-i}$$

où i est le rang de la *p-value* après classement.

La technique du False Discovery Rate (FDR) de Benjamini et Hochberg

La procédure se réalise en plusieurs étapes :

1. Classer les *p-values* de tous les tests réalisés par ordre croissant ($p_1 < \ldots < p_k$), k étant le nombre de tests effectués.
2. Rejeter H_0 pour les tests dont la *p-value* satisfait la condition :

$$p_i \leq \alpha_{\text{seuil}} \times \frac{i}{k}$$

où i est le rang de la *p-value* après classement.

La technique la plus stricte est celle de Bonferroni, la moins stricte celle du FDR. Cette dernière peut être appliquée par défaut. Dans tous les cas la méthode de correction du seuil de rejet de H_0 doit être décidée *avant de réaliser les tests.*

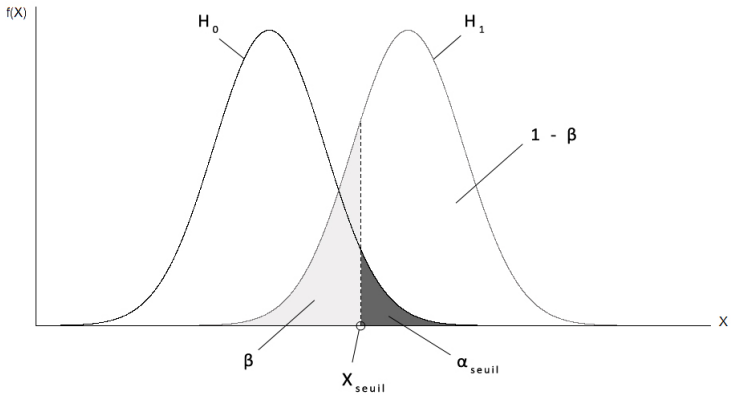
Dans **R**, si `p` est un vecteur contenant les *p-values* non corrigées, utiliser la fonction `p.adjust()` pour récupérer un vecteur avec les *p-values* corrigées (dans le même ordre) :
`p.adjust(p,method="bonferroni")` pour la correction de Bonferroni
`p.adjust(p,method="holm")` pour la correction de Holm
`p.adjust(p,method="BH")` ou `p.adjust(p,method="fdr")` pour la correction de Benjamini et Hochberg (FDR).

17. Risque β et puissance d'un test

Le risque β est le risque de ne pas rejeter l'hypothèse H_0 *si celle-ci est réellement fausse* (voir fiche **14**). Contrairement au risque α, le risque β ne peut pas être fixé. En effet, si α dépend de la distribution de la Variable de Test (VT) sous H_0 (voir fiche **15**), β dépend de sa distribution sous H_1. Or cette distribution est inconnue, puisque l'hypothèse H_1 regroupe une infinité de distributions (ex : si l'hypothèse H_1 est $\mu_A \neq \mu_B$, les deux moyennes μ_A et μ_B peuvent différer d'une infinité de façons).

La puissance d'un test représente la probabilité de rejeter H_0 *si celle-ci est réellement fausse* (*i.e.* de faire le bon choix). Elle équivaut à $1-\beta$, et est donc également une variable dépendant de la distribution de la VT sous H_1.

Le risque β et la puissance du test dépendent du seuil α fixé :



X_{seuil} : valeur de la VT X qui donne une fonction de répartition à droite égale au seuil α pour la distribution sous H_0 (test unilatéral droit ici).

La puissance d'un test augmente :
- quand augmente le seuil α
- quand augmente l'effectif de l'échantillon testé (ce qui diminue l'étalement de la distribution de la VT ou éloigne les distributions de la VT sous H_0 et H_1, selon le test)
- quand augmente l'écart réel entre les valeurs testées (moyennes, proportions...).

18. Déterminer la taille de l'échantillon à constituer

— PACKAGE(S) ADDITIONNEL(S) UTILISÉ(S) DANS CETTE FICHE —

[1]pwr

Il existe un lien entre le seuil de rejet α du test statistique utilisé (voir fiches **14** et **15**), la puissance de ce test (voir fiche **17**), la différence entre les échantillons pour la variable mesurée et la taille des échantillons. Déterminer la taille de l'échantillon à constituer passe donc par fixer les autres paramètres. Ceci implique deux choses importantes :

- choisir *avant de démarrer l'étude* les types de test qui vont être utilisés (ce qui oblige à bien identifier les questions auxquelles l'étude doit répondre) et leur précision. On considère que lorsque l'objectif de l'étude est de *rejeter* H_0 (voir fiche **14**), la puissance du test doit être d'au moins 80 % ; lorsque l'objectif est de *ne pas rejeter* H_0 (voir fiche **14**), la puissance doit être d'au moins 95 %.
- avoir une idée de la variabilité naturelle de la variable mesurée et/ou de la différence minimale à détecter (le seuil de significativité *biologique*, voir fiche **14**). Ceci passe soit par une étude de la bibliographie, soit par la consultation de spécialistes, soit par la réalisation d'un pré-échantillonnage ou d'une pré-expérience.

Dans **R**, la fonctions `power()` permet, pour quelques tests paramétriques courants, de calculer l'effectif nécessaire lorsque l'on connaît tous les autres paramètres. Lorsque l'on sait à l'avance que l'on ne sera pas dans les conditions d'utilisation de ces tests, on peut tout de même estimer la puissance de leur équivalent non paramétrique. Celle-ci est vaut en effet environ 95 % de la puissance du test paramétrique équivalent.

Toutes les fonctions décrites sont basées sur le même principe : le paramètre `n` doit avoir comme valeur `NULL` tandis que tous les autres doivent être fixés. Toutes les fonctions considèrent que l'effectif est le même dans les différents groupes à comparer (on parle d'effectifs *équilibrés*). Il est fortement conseillé de toujours prévoir ses plans d'échantillonnage ou d'expérience de cette façon, ce qui (i) facilite l'analyse et (ii) permet d'utiliser la plupart des tests non paramétriques, dont l'équilibre des effectifs est une condition.

Comparaison d'une moyenne avec une valeur théorique (voir fiche 72) ou de deux moyennes (voir fiche 73) – test *t* de Student

```
power.t.test(n,delta,sd,sig.level,power,type)
```

avec :
`n` : effectif (identique pour les deux échantillons dans le cas d'une comparaison de deux moyennes)
`delta` : différence minimale à détecter entre la moyenne de l'échantillon et la moyenne théorique, ou entre les deux moyennes
`sd` : écart-type (identique pour les deux échantillons dans le cas d'une comparaison de deux moyennes), dans l'unité des moyennes
`sig.level` : seuil de rejet α (généralement 0,05)
`power` : puissance minimale du test (0,8 ou 0,95 selon l'objectif du test)
`type` : type de test (`"one.sample"` pour la comparaison d'une moyenne avec une valeur théorique, `"two.sample"` pour la comparaison de deux moyennes, `"paired"` pour la comparaison de deux moyennes en séries appariées).

Comparaison de plus de deux moyennes – analyse de variance à un facteur (voir fiche 76)

```
power.anova.test(groups,n,between.var,within.var,sig.level,power)
```

avec :
`groups` : nombre de modalités à comparer
`between.var` : somme des carrés des écarts intergroupe minimale à détecter
`within.var` : somme des carrés des écarts intragroupe (identique pour toutes les modalités).

Il est dans la pratique très difficile d'estimer *a priori* les sommes des carrés des écarts inter et intragroupe. On peut dans ce cas se rabattre sur la fonction `power.t.test()` qui donne des résultats convenables si les conditions d'utilisation de l'analyse de variance sont remplies.

Comparaison de deux proportions – test du χ^2 d'homogénéité (voir fiche 58)

`power.prop.test(n,p1,p2,sig.level,power)`

avec `p1`, `p2` : proportions à détecter comme significativement différentes.

Significativité d'un coefficient de corrélation linéaire de Pearson (voir fiche 84)

`pwr.r.test(n,r,sig.level,power)`[1]

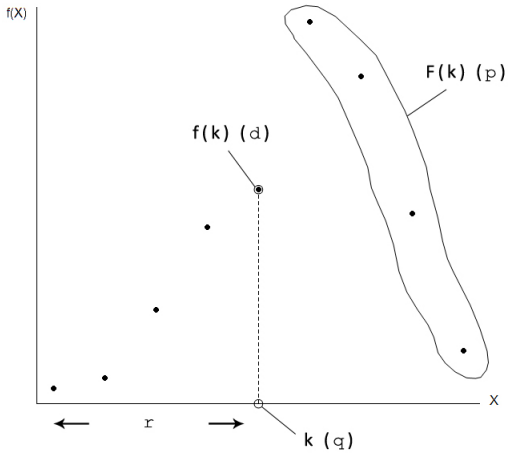
avec `r` : coefficient de corrélation minimum à détecter comme significativement différent de 0.

19. Les lois de probabilité discontinues – généralités

Ces lois s'appliquent à des variables quantitatives discrètes (voir fiche **11**).

Leurs paramètres sont :
— k : chaque valeur possible rencontrée dans la population par la variable discrète X. Également appelée « quantile ».
— $f(k)$: fréquence, ou probabilité, associée à chaque valeur de la variable discrète X. Également appelée « distribution de probabilité » de X ou « fonction de masse » de X. Comprise entre 0 et 1.
— $F(k)$: somme des probabilités $f(k)$ situées à droite ou à gauche de k, suivant la situation. Également appelée « fonction de répartition » de X. On note $F(k)_{\text{droite}} = P(X > k)$ et $F(k)_{\text{gauche}} = P(X \leq k)$. Comprise entre 0 et 1.

Dans **R** :
— `dY()` : donne la probabilité $f(k)$ pour une distribution de type `Y`.
— `pY()` : donne la fonction de répartition $F(k)$ pour une distribution de type `Y`. **R** considère par défaut la répartition à gauche, préciser `lower.tail=FALSE` pour la répartition à droite.
— `qY()` : donne la valeur k de la variable X correspondant à une valeur de $F(k)$ pour une distribution de type `Y`. **R** considère par défaut la répartition à gauche de k, préciser `lower.tail=FALSE` pour la répartition à droite.
— `rY()` : donne une série de valeurs aléatoires de la variable X pour une distribution de type `Y`.

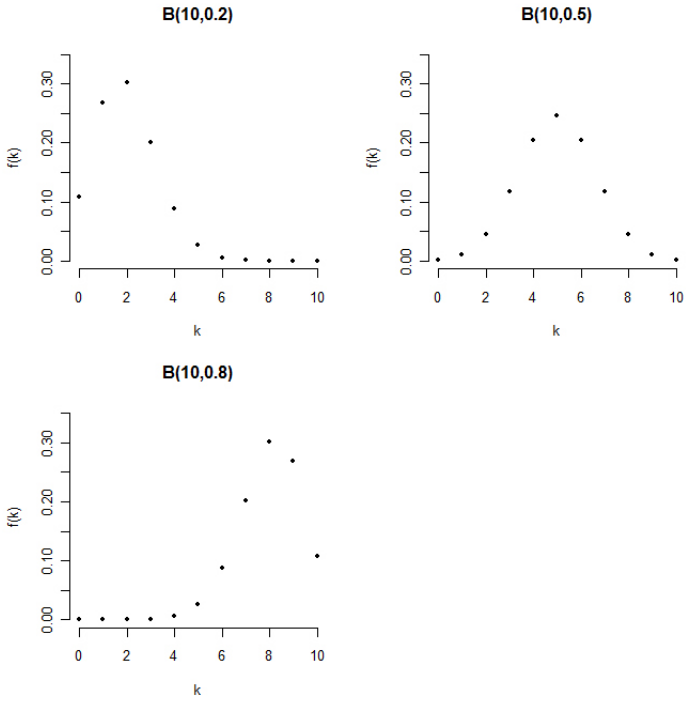


20. La loi binomiale

La loi binomiale est la loi suivie par les résultats de tirages aléatoires lorsqu'il n'y a que deux possibilités mutuellement exclusives de résultat et que la probabilité d'obtenir chaque possibilité est constante au cours de l'expérience (*i.e.* population infinie ou tirages avec remise). La loi donne la probabilité d'obtenir k fois le résultat A quand n tirages sont réalisés.

Écriture : $B(n,p)$
avec :
n : nombre de tirages
p : probabilité associée au résultat A

Dans **R** (voir fiche **19** pour des explications) :
`dbinom(k,n,p)`
`pbinom(k,n,p)`
`qbinom(F(k),n,p)`
`rbinom(x,n,p)` avec `x` : nombre de valeurs à générer



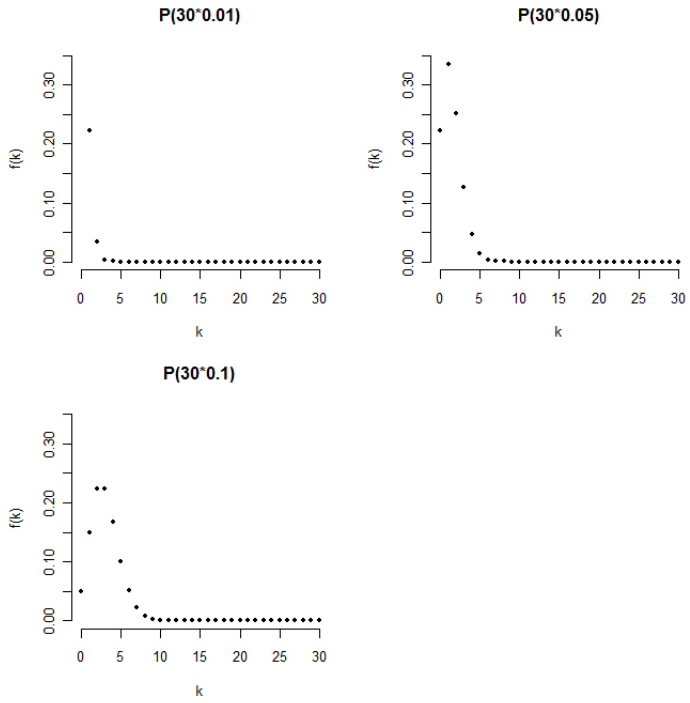
21. La loi de Poisson

La loi de Poisson est une limite de la loi binomiale (voir fiche **20**) quand p tend vers 0 et n vers l'infini (la loi de Poisson est souvent appelée « loi des évènements rares »). L'approximation de la loi binomiale par la loi de Poisson est possible quand $p < 0,1$ et $n > 30$.

Sa moyenne est égale à sa variance et vaut np (ou λ).

Écriture : $P(np)$ ou $P(\lambda)$
avec :
n : nombre de tirages
p : probabilité associée au résultat rare

Dans **R** (voir fiche **19** pour des explications) :
`dpois(k,n*p)` ou `dpois(k,lambda)`
`ppois(k,n*p)` ou `ppois(k,lambda)`
`qpois(F(k),n*p)` ou `qpois(F(k),lambda)`
`rpois(x,n*p)` ou `rpois(x,lambda)` avec `x` : nombre de valeurs à générer

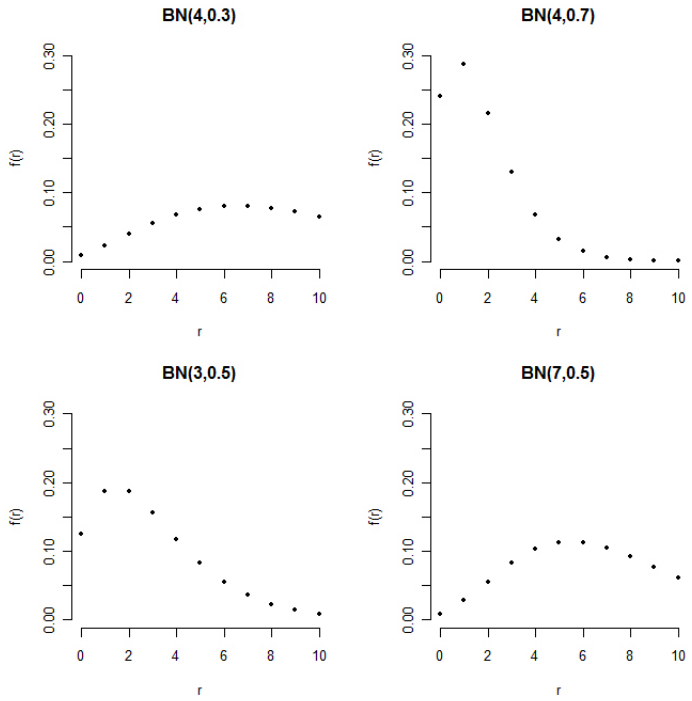


22. La loi binomiale négative

La loi binomiale négative correspond à la même situation que la loi binomiale (voir fiche **20**), mais elle donne la probabilité d'obtenir r résultats B avant d'obtenir k résultats A (approche par l'échec).

Écriture : $BN(k,p)$
avec :
k : nombre de résultats A désirés
p : probabilité associée au résultat A

Dans **R** (voir fiche **19** pour des explications) :
`dnbinom(r,k,p)`
`pnbinom(r,k,p)`
`qnbinom(F(r),k,p)`
`rnbinom(x,k,p)` avec `x` : nombre de valeurs à générer

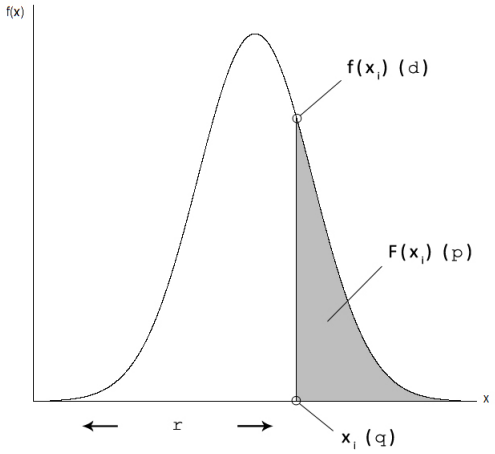


23. Les lois de probabilité continues – généralités

Ces lois s'appliquent à des variables quantitatives continues (voir fiche **11**).

Leurs paramètres sont :
— x_i : chaque valeur possible de la variable continue x. Également appelée « quantile ».
— $f(x_i)$: distribution de probabilité de la valeur x_i. Également appelée « densité de probabilité » de x_i. Comprise entre 0 et 1.
— $F(x_i)$: aire sous la courbe située à droite ou à gauche de x_i, suivant la situation. Également appelée « fonction de répartition » de x_i. On note $F(x_i)_{\text{droite}} = P(x > x_i)$ et $F(x_i)_{\text{gauche}} = P(x \leq x_i)$. Comprise entre 0 et 1.

Dans **R** :
— `dY()` : donne la densité de probabilité $f(x_i)$ pour une distribution de type `Y`.
— `pY()` : donne la fonction de répartition $F(x_i)$ pour une distribution de type `Y`. **R** considère par défaut la répartition à gauche, préciser `lower.tail=FALSE` pour la répartition à droite.
— `qY()` : donne la valeur x_i de la variable x correspondant à une valeur de $F(x_i)$ pour une distribution de type `Y`. **R** considère par défaut la répartition à gauche de k, préciser `lower.tail=FALSE` pour la répartition à droite.
— `rY()` : donne une série de valeurs aléatoires de la variable x pour une distribution de type `Y`.



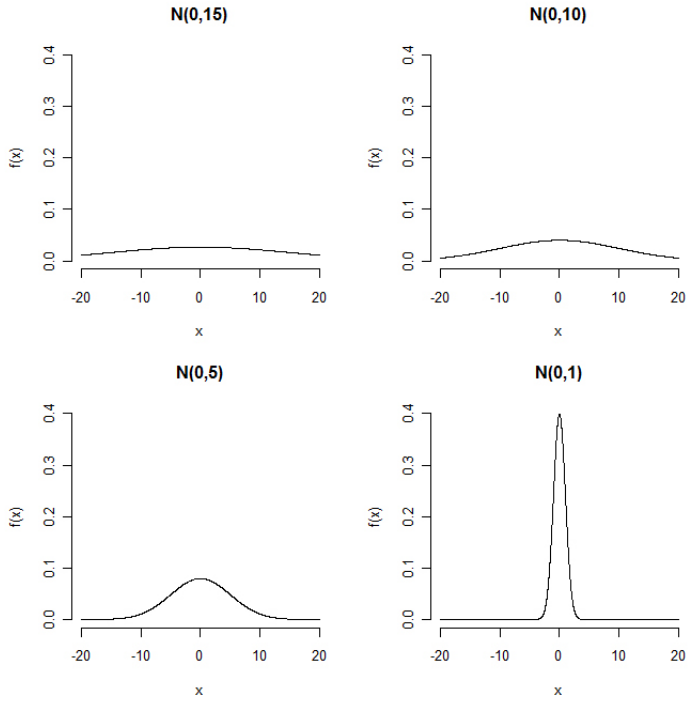
24. La loi normale

La loi normale est la plus fondamentale de toutes les stastiques puisqu'elle sert à modéliser un bruit aléatoire autour une moyenne. Elle est donc le support de nombreux modèles et tests, comme condition préalable à leur utilisation et/ou comme distribution de référence pour calculer la *p-value*.

Écriture : $N(\mu, \sigma)$
avec :
μ : moyenne de la variable x
σ : écart-type de la variable x

Cas particulier, la loi normale centrée-réduite : $N(0,1)$

Dans **R** (voir fiche **23** pour des explications) :
`dnorm(xi,mu,sigma)`
`pnorm(xi,mu,sigma)`
`qnorm(F(xi),mu,sigma)`
`rnorm(z,mu,sigma)` avec `z` : nombre de valeurs à générer

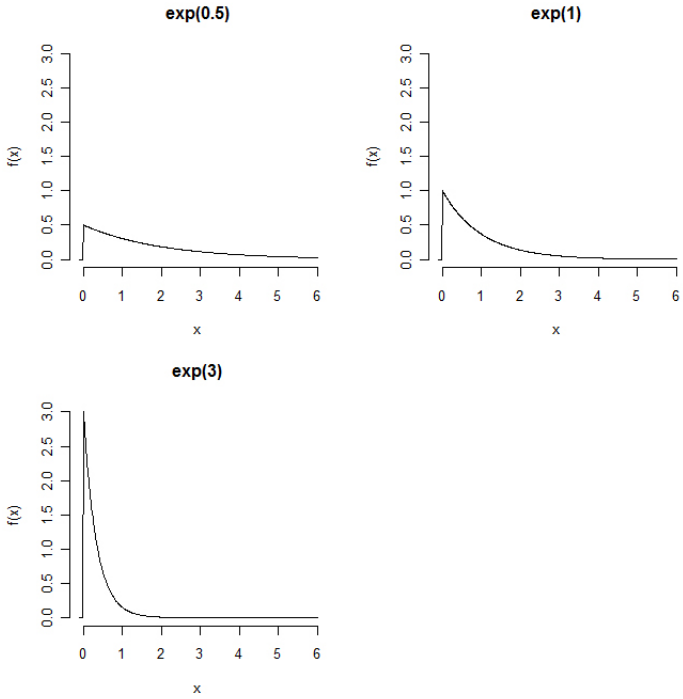


25. La loi exponentielle

La loi exponentielle correspond souvent à des évènements dont la probabilité de survenue diminue avec le temps. Elle est donc souvent utilisée pour modéliser des durées de vie.

Écriture : $exp(\lambda)$
avec λ : paramètre de la loi $(0 < \lambda < +\infty)$

Dans **R** (voir fiche **23** pour des explications) :
`dexp(xi,lambda)`
`pexp(xi,lambda)`
`qexp(F(xi),lambda)`
`rexp(z,lambda)` avec `z` : nombre de valeurs à générer

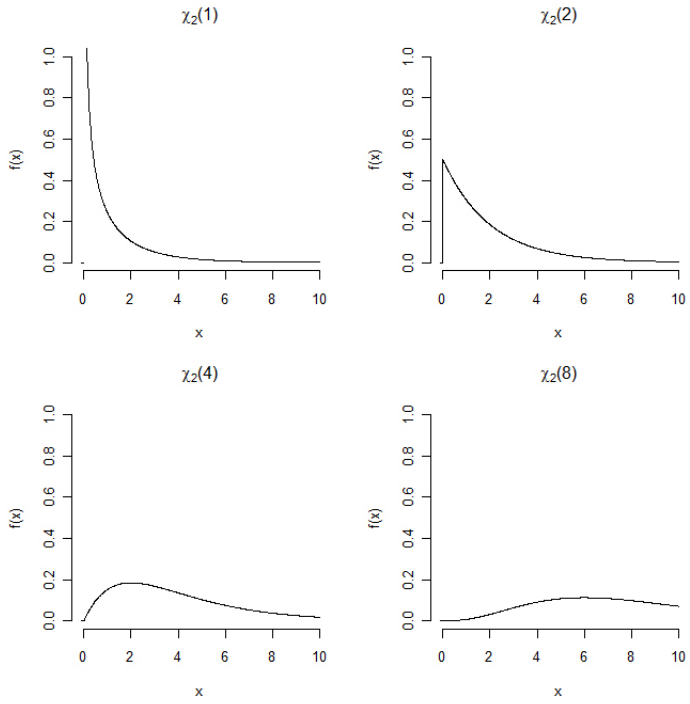


26. La loi de χ^2

La loi de χ^2 est utilisée dans de nombreux tests statistiques (« tests du χ^2 » mais pas seulement) comme distribution de référence pour calculer la *p-value*.

Écriture : $\chi^2(\nu)$
avec ν : nombre de degrés de liberté (ddl), *i.e.* de paramètres indépendants impliqués dans la loi ($0 < \nu < +\infty$)

Dans **R** (voir fiche **23** pour des explications) :
`dchisq(xi,ddl)`
`pchisq(xi,ddl)`
`qchisq(F(xi),ddl)`
`rchisq(z,ddl)` avec `z` : nombre de valeurs à générer

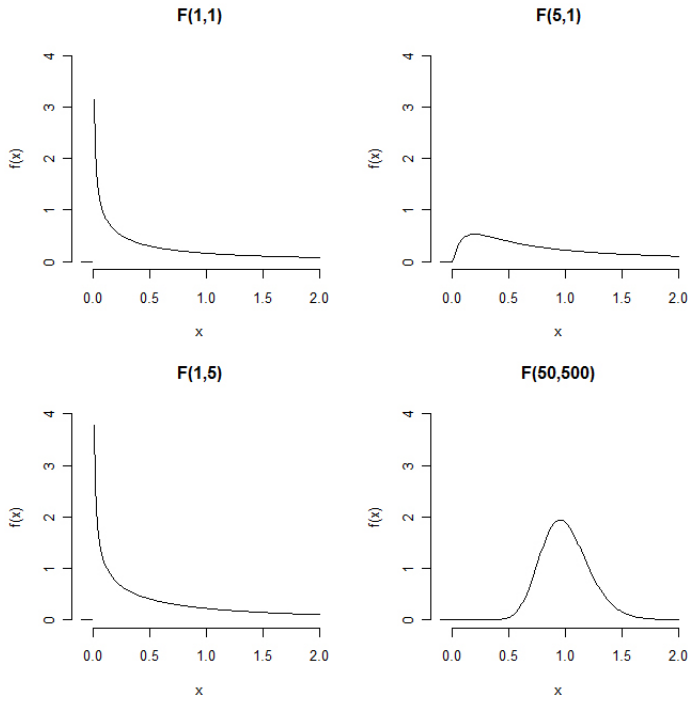


27. La loi de Fisher - Snedecor

La loi de Fisher-Snedecor est utilisée dans de nombreux tests statistiques (ANOVA mais pas seulement) comme distribution de référence pour calculer la *p-value*.

Écriture : $F(\nu_1, \nu_2)$
avec :
ν_1 : 1^er^ nombre de degrés de liberté (ddl) ($0 < \nu_1 < +\infty$)
ν_2 : 2^ème^ nombre de ddl ($0 < \nu_2 < +\infty$)

Dans **R** (voir fiche **23** pour des explications) :
`df(xi,ddl1,ddl2)`
`pf(xi,ddl1,ddl2)`
`qf(F(xi),ddl1,ddl2)`
`rf(z,ddl1,ddl2)` avec `z` : nombre de valeurs à générer

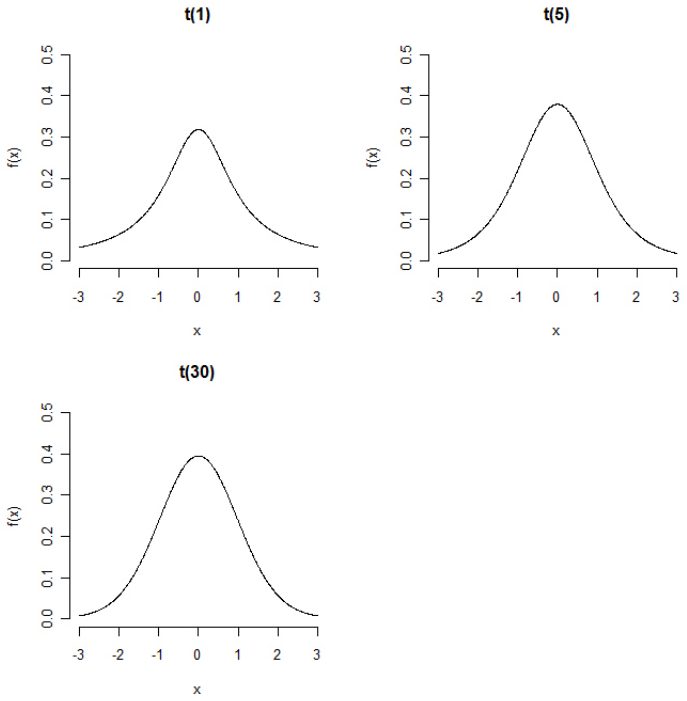


28. La loi de Student

La loi de Student est utilisée dans de nombreux tests statistiques (test t de Student mais pas seulement) comme distribution de référence pour calculer la *p-value*.

Écriture : $t(\nu)$
avec ν : nombre de degrés de liberté (ddl) $(0 < \nu < +\infty)$

Dans **R** (voir fiche **23** pour des explications) :
`dt(xi,ddl)`
`pt(xi,ddl)`
`qt(F(xi),ddl)`
`rt(z,ddl)` avec `z` : nombre de valeurs à générer



Troisième partie

Analyser les résultats d'une étude

La première question à se poser en terme d'analyse statistique est le cadre général dans lequel on veut se placer : univarié, bivarié ou multivarié. Il est en pratique assez simple de s'orienter :

— si l'on a *une variable à expliquer*, dont on cherche à savoir si ses valeurs sont influencées par une ou plusieurs variable(s) explicative(s) : statistique *univariée*.
— si l'on étudie la relation entre *deux variables du même type* (quantitatives ou qualitatives), sans qu'il y ait de variable à expliquer et de variable explicative : statistique *bivariée*. On parle dans ce cas de variables *interdépendantes*.
— si l'on a *plusieurs variables à expliquer conjointement* : statistique *multivariée*.

Lorsque plusieurs variables sont à expliquer conjointement, les approches multivariées (*i.e.* globale) et univariées (*i.e.* chaque variable séparément) peuvent être complémentaires lorsque les deux sont possibles. L'approche multivariée devient surtout intéressante quand les variables sont corrélées entre elles, car elle prend en compte cette corrélation et permet de mettre en évidence des phénomènes qui ne sont visibles que d'un point de vue global (et pas à l'échelle des variables individuelles). Il n'existe pas de seuil de corrélation absolu pour estimer l'intérêt de l'approche multivariée ; cela ne coûte de rien d'essayer et de voir si un phénomère intéressant apparaît !

Note : souvent, plusieurs tests peuvent être utilisés pour répondre à la même question. Les conditions de leur emploi sont cependant plus ou moins restrictives, et leur puissance plus ou moins grande (un test plus restrictif étant généralement plus puissant ; voir fiche **17**). Lorsque plusieurs tests sont disponibles, un arbre de décision est présenté pour aider à choisir le test approprié. Il s'appuie à la fois sur la vérification des conditions à remplir pour utiliser tel ou tel test, ainsi que sur la « qualité » de ces tests (notamment en terme de puissance).

Données à une dimension – Statistique univariée

GRAPHES

30. Graphiques de dispersion
31. Histogrammes
32. Boîtes à moustaches
33. Graphiques en haricots
34. Diagrammes en barres avec barres d'erreur

RÉDUCTION DES DONNÉES

35. Paramètres de position (moyenne, médiane, mode)
36. Paramètres de dispersion (variance, écart-type, coefficient de variation)
37. Intervalle de confiance et erreur standard (d'une moyenne, d'une médiane, d'une proportion)

TESTS

38. Identification et suppression des données aberrantes

Comment s'orienter dans la jungle des tests et des modèles ?

Pour s'y retrouver, il faut oublier le vocabulaire habituel (« moyennes », « proportions », « effectifs »…) et se concentrer uniquement sur la *nature de la variable à expliquer*, peu importe comment elle est codée. Les situations suivantes sont traitées dans cet aide-mémoire (voir fiche **11** si les termes ne sont pas clairs) :

1. **La réponse est qualitative**
   (a) nominale :
      i. à 2 classes (*i.e.* binaire)
      ii. à plus de 2 classes
   (b) ordinale (*i.e.* un rang dans un classement)
2. **La réponse est quantitative**
   (a) discontinue (*i.e.* un décompte d'individus) :
      i. sans catégorie
      ii. par catégorie :
         A. dans 2 catégories
         B. dans plus de 2 catégories
   (b) continue (théoriquement, pas forcément dans les faits) :
      i. bornée (dans les faits), *i.e.* plusieurs valeurs sont aux limites des valeurs possibles (par exemple 0 et/ou 1 pour une proportion)
      ii. non bornée (dans les faits), *i.e.* même si toutes les valeurs ne sont pas possibles les données sont (quasiment) toutes entre les limites sans les atteindre

   Remarque : si la réponse est un temps avant la survenue d'un évènement, elle est traitée à part grâce à des méthodes dédiées.

De manière générale, les variables explicatives doivent être bien moins nombreuses que les individus (pour faire simple : 10 fois moins nombreuses) et elles ne doivent pas être trop corrélées entre elles (pour faire simple : sans relation significative).

Les tests proposés sont systématiquement divisés en deux groupes : les tests « simples » (essentiellement non paramétriques), qui exigent peu de connaissances en statistiques pour être utilisés ; les tests

paramétriques basés sur un modèle statistique, qui font appel à des notions un peu plus avancées (tout en restant appréhendables, c'est le but de ce document que de rendre les choses simples !). Les modèles sont objectivement plus puissants et permettent bien plus de choses que les tests non paramétriques, et à ce titre ils sont l'approche à privilégier *a priori* lorsque plusieurs alternatives sont disponibles. On pourra toutefois s'orienter vers les tests simples si les conditions d'utilisation du modèle ne sont pas respectées, ou si l'on souhaite délibérément utiliser ce type de test.

La gestion des modèles

Les notions des fiches suivantes (surtout **39** à **42**) sont indispensables à maîtriser pour utiliser proprement un modèle.

39. Démarche d'utilisation des modèles
40. Construction de la formule d'un modèle
41. Vérification de la validité d'un modèle
42. Application d'un test sur un modèle
43. Comparaisons multiples basées sur un modèle
44. Sélection de modèle

La réponse est binaire

☛ Tests simples

Par commodité on va appeler la réponse « probabilité » dans les fiches suivantes. Mais en fait « probabilité » ne se résume pas à ce type de réponse.

45. Conformité d'une probabilité à une valeur théorique
46. Conformité de plusieurs probabilités à des valeurs théoriques – 2 classes
47. Comparaison de plusieurs probabilités – 2 classes

☛ Tests basés sur un modèle

48. Analyser une réponse binaire

La réponse est qualitative nominale à plus de 2 classes

☛ Tests simples

Par commodité on va appeler la réponse « probabilité » dans les fiches suivantes. Mais en fait « probabilité » ne se résume pas à ce type de réponse.

49. Conformité de plusieurs probabilités à des valeurs théoriques – plus de 2 classes
50. Comparaison de plusieurs probabilités – plus de 2 classes

☛ Tests basés sur un modèle

51. Analyser une réponse qualitative nominale à plus de 2 classes

La réponse est un rang dans un classement

Il est préférable d'éviter d'utiliser les tests simples non paramétriques sur les rangs (Mann-Whitney-Wilcoxon, Kruskal-Wallis, Friedman...) pour analyser de telles variables. En effet elles contiennent souvent de nombreuses valeurs identiques (car seules quelques valeurs sont possibles quand on travaille sur un classement), ce qui biaise complètement ces tests.

☛ Tests basés sur un modèle

52. Analyser un rang dans un classement

La réponse est un décompte d'individus sans catégorie

☛ Tests simples

Par commodité on va appeler la réponse « effectif » dans les fiches suivantes. Mais en fait « effectif » ne se résume pas à ce type de réponse.

53. Conformité d'une série d'effectifs à une distribution théorique
54. Comparaison de plusieurs effectifs

☛ Tests basés sur un modèle

55. Analyser un décompte d'individus

La réponse est un décompte d'individus dans 2 catégories

☛ Tests simples

Par commodité on va appeler la réponse « proportion » dans les fiches suivantes. Mais en fait « proportion » ne se résume pas à ce type de réponse.

56. Conformité d'une proportion à une valeur théorique
57. Conformité de plusieurs proportions à des valeurs théoriques – 2 catégories
58. Comparaison de deux proportions – 2 catégories
59. Comparaison de plus de deux proportions – 2 catégories

☛ Tests basés sur un modèle

60. Analyser un décompte d'individus dans 2 catégories

La réponse est un décompte d'individus dans plus de 2 catégories

☛ Tests simples

Par commodité on va appeler la réponse « proportion » dans les fiches suivantes. Mais en fait « proportion » ne se résume pas à ce type de réponse.

61. Conformité de plusieurs proportions à des valeurs théoriques – plus de 2 catégories
62. Comparaison de plusieurs proportions – plus de 2 catégories

☛ Tests basés sur un modèle

63. Analyser un décompte d'individus dans plus de 2 catégories

La réponse est continue bornée

64. Analyser une réponse continue bornée

La réponse est continue non bornée

☛ Tests simples portant sur la distribution

65. Conformité d'une variable continue à une distribution théorique
66. Comparaison de deux distributions

☛ Tests simples portant sur la variance

67. Comparaison de deux variances
68. Comparaison de plus de deux variances

☛ Tests simples portant sur la médiane

69. Conformité d'une médiane à une valeur théorique
70. Comparaison de deux médianes
71. Comparaison de plus de deux médianes

☛ Tests simples portant sur la moyenne

72. Conformité d'une moyenne à une valeur théorique
73. Comparaison de deux moyennes
74. Comparaison de plus de deux moyennes – 1 facteur
75. Comparaison de plus de deux moyennes – 2 facteurs

Remarque : l'ANOVA est en fait un test basé sur un modèle. Elle est donc présentée dans la fiche **76**.

☛ Tests basés sur un modèle

76. Analyser une variable continue non bornée – relation(s) linéaire(s)
77. Analyser une variable continue non bornée – relation non linéaire

La réponse est un temps avant la survenue d'un évènement

☛ Tests basés sur un modèle

Par commodité on va appeler un temps avant la survenue d'un évènement « temps de survie » dans les fiches suivantes (et l'évènement lui-même « mort »), car les modèles développés pour ce type de réponse l'ont originellement été pour analyser des temps de survie. Ce n'est qu'une question de vocabulaire, la démarche est la même quel que soit l'évènement.

78. Choisir le modèle d'analyse d'un temps de survie
79. Analyser un temps de survie – Modèle Linéaire Généralisé
80. Analyser un temps de survie – Régression de survie
81. Analyser un temps de survie – Modèle de Cox

Données à deux dimensions – Statistique bivariée

GRAPHES

82. Nuages de points

RÉDUCTION DES DONNÉES

83. Intensité de la liaison entre deux variables (covariance, corrélation, association)

TESTS

☛ Tests simples

84. Corrélation entre deux variables quantitatives ou ordinales
85. Comparaison de plusieurs coefficients de corrélation
86. Association entre deux variables qualitatives

☛ Tests basés sur un modèle

87. Analyser deux variables quantitatives interdépendantes

Données à plus de deux dimensions – Statistique multivariée

Comment s'orienter dans la jungle des analyses multivariées ?

Pour s'y retrouver, il faut se poser quelques questions très simples. La première est : combien de jeux de données (ou « tableaux ») y a-t-il à analyser ?

1. Un seul. L'information utilisée dans l'analyse est donc uniquement celle contenue dans ce jeu de données.
   Remarque : s'il y a la moindre autre variable à considérer que celles du jeu de données lui-même (des groupes connus à l'avance par exemple), il y a *deux* jeux de données. Et ce même si le second ne contient qu'une seule variable.
   Question suivante : quel est l'objectif de l'analyse ?
   (a) Classer les individus en groupes. On utilise alors une méthode de *classification*, qui va constituer les groupes de façon objective.
   (b) Synthétiser l'information contenue dans le jeu de données pour la visualiser et l'interpréter. On utilise alors une méthode d'*ordination*.
   Remarque : on peut toujours tester si des variables « externes », *i.e.* connues à l'avance mais non prises en compte dans l'analyse, sont significativement « corrélées » au résultat d'une ordination (voir fiche **91**).
2. Deux. On parle d'analyse *canonique*.
   Question suivante : quelle est la relation entre les deux jeux de données ?
   (a) L'un est considéré comme expliquant l'autre, *i.e.* un jeu de données est « explicatif » et l'autre « à expliquer ». On utilise alors une analyse *asymétrique*. Comme pour tout modèle (univarié ou multivarié), les variables explicatives doivent être bien moins nombreuses que les individus et elles ne doivent pas être trop corrélées entre elles.
   (b) Les deux sont considérés de la même façon, sur un même pied d'égalité. On utilise alors une analyse *symétrique*.
   Remarque : certains auteurs utilisent le terme « canonique » uniquement pour les analyses symétriques.
3. Plus de deux. On est là encore dans le cadre des analyses canoniques. Les analyses présentées dans ce document sont toutes symétriques.

Trois remarques :

— Pour certaines analyses, il peut être intéressant voire nécessaire de pré-traiter les données en amont (voir fiche **88**). Ces cas sont identifiés dans les fiches concernées.
— Quelle que soit l'analyse, le « jeu de données » peut être constitué soit des variables d'origine (pré-traitées ou non), soit d'une matrice de distance (voir fiche **101**), soit des variables de synthèse obtenues par une méthode d'ordination (*i.e.* les coordonnées des individus sur les axes créés par une précédente analyse, voir fiche **90**). Les cas où cette dernière démarche peut être intéressante sont identifiées dans les fiches concernées.
— Lorsque plusieurs tableaux sont analysés en même temps leurs lignes doivent être *identiques*, *i.e.* les différents tableaux sont des ensembles de variables mesurées sur *les mêmes individus*.

DIVERS

ANALYSES PORTANT SUR 1 TABLEAU

Classification

Approches possibles en classification

La classification vise toujours à regrouper objectivement les individus, dans des groupes qui sont à la fois les plus homogènes possibles à l'intérieur et les plus hétérogènes possibles entre eux. En pratique il existe cependant deux manières d'aborder la classification, qui diffèrent par leurs objectifs :

1. soit on cherche à regrouper les individus dans des groupes emboîtés (*i.e.* savoir « qui est plus proche de qui »), dont on veut éventuellement tester la solidité (*i.e.* la « significativité »)
2. soit on cherche à définir le nombre optimal de groupes qui structurent les données, et à assigner de manière optimale chaque individu à un groupe.

☛ Classifier pour obtenir une hiérachie de groupes emboîtés

92. Classification – Identifier et tester une hiérarchie

☛ Classifier pour identifier le nombre optimal de groupes

Pour être réalisée correctement, cette approche se fait en quatre étapes, dans l'ordre suivant :

93. Classification – 1. Tester si un jeu de données peut être classifié
94. Classification – 2. Identifier le nombre optimal de groupes
95. Classification – 3. Réaliser la classification
96. Classification – 4. Valider le résultat d'une classification

Ordination

Une ordination peut se faire directement sur un tableau de variables, ou sur une *matrice de distance*. Une telle matrice contient toutes les distances deux-à-deux entre les individus, qui peuvent avoir été calculées à partir de variables ou avoir été obtenues directement (distances génétiques par exemple). Il n'y a donc plus aucune information sur les éventuelles variables d'origine dans une matrice de distance. Choisir entre les deux types d'analyse est relativement simple : si la question porte (donc si l'interprétation biologique repose) sur les variables, utiliser une analyse sur les variables. Si la question porte sur la (dis)similarité globale entre individus, utiliser une analyse sur une matrice de distance.

☛ Analyses sur les variables

Ordination sur un tableau de variables

Le choix de l'analyse dépend de la nature du tableau et des variables qu'il contient :

1. Le tableau est un tableau de variables classique (*i.e.* individus en lignes, variables en colonnes). Ces variables sont :
   (a) toutes quantitatives → Analyse en composantes principales
   (b) toutes qualitatives nominales (*i.e.* non ordonnées). Le choix dépend alors du nombre de variables :
      i. Deux → réorganiser le tableau en tableau de contingence (voir 2.) puis Analyse factorielle des correspondances
      ii. Plus de deux → Analyse des correspondances multiples
   (c) de plusieurs types (quantitatives, qualitatives nominales, qualitatives ordinales) ou toutes qualitatives ordinales → Analyse mixte
2. Le tableau est un croisement des modalités de deux facteurs, où chaque case contient un nombre d'individus (le tableau est alors appelé *tableau de contingence*) ou une réponse binaire (tableau de présence-absence) → Analyse factorielle des correspondances.

97. L'analyse en composantes principales (ACP ou *PCA*)
98. L'analyse factorielle des correspondances (AFC ou *CA*)
99. L'analyse des correspondances multiples (ACM ou *MCA*)
100. L'analyse mixte

☛ Analyses sur une matrice de distance

Ordination sur une matrice de distance

L'analyse en coordonnées principales (PCoA) et le positionnement multidimensionnel non métrique (nMDS) ont le même objectif : créer une ordination à partir d'une matrice de distance. Mais dans le détail, deux différences les distinguent :

1. La PCoA préserve les distances réelles, tandis que la nMDS n'utilise qu'une information semi-quantitative (les rangs). Autrement dit, les distances observées sur une ordination réalisée par PCoA peuvent être directement interprétées comme les distances qui séparent les individus dans la matrice de distance ; sur une ordination réalisée par nMDS, la seule information disponible est du type « telle distance est plus grande que telle autre », car les distances réelles ne sont pas préservées.

2. Comme les autres méthodes d'ordination, la PCoA produit de nombreux axes et l'on retient les premiers pour l'interprétation (donc seulement une partie de l'information, l'autre étant sur les axes non retenus) ; la nMDS produit une ordination sur un nombre d'axes déterminé *a priori* (2 ou 3 en pratique), donc toute l'information est facilement interprétable.

Si l'on souhaite garder l'information quantitative sur les distances réelles, en interprétant toutefois sur une information partielle, choisir la PCoA. Si l'on souhaite synthétiser toute l'information pour l'interprétation, en perdant l'information quantitative sur les distances réelles, choisir la nMDS.
Remarque : si l'on prévoit de « corréler » les résultats de l'ordination à des variables externes (voir fiche **91**), seule la PCoA a réellement du sens.

101. Les matrices de distance
102. L'analyse en coordonnées principales (*PCoA*)
103. Le positionnement multidimensionnel non métrique (*nMDS*)

ANALYSES PORTANT SUR 2 TABLEAUX

Analyses asymétriques

☛ Analyses sur les variables

Analyses asymétriques sur un tableau de variables

Le choix de l'analyse dépend de l'objectif, de la nature des variables du tableau à expliquer, et de la nature des variables du tableau explicatif :

1. L'objectif est de réaliser une ordination. Le tableau explicatif est constitué :
   (a) D'une seule variable qualitative (*i.e.* un facteur définissant des groupes). On parle alors de *discrimination*. Le tableau à expliquer est :
      i. Un tableau de variables quantitatives. Ces variables sont :
         A. bien moins nombreuses que les individus (minimum 5 fois moins) et pas (ou peu) corrélées entre elles → Analyse discriminante linéaire
         B. aussi nombreuses voire plus nombreuses que les individus et/ou corrélées entre elles → Régression PLS discriminante
      ii. Un tableau de contingence ou tableau de présence-absence → Analyse des correspondances discriminante
   (b) De plusieurs variables (quantitatives et/ou qualitatives), ou d'une seule variable quantitative. Le tableau à expliquer est :
      i. Un tableau de variables quantitatives → Analyse de redondance
      ii. Un tableau de contingence ou tableau de présence-absence → Analyse canonique des correspondances

   Remarque : si le tableau de variables ne rentre pas dans les cases précédentes, le transformer en matrice de distance (voir fiche **101**) et utiliser l'analyse de redondance sur matrice de distance.

2. L'objectif est simplement de réaliser un test statistique → Analyser un ensemble de variables quantitatives. Les variables du tableau à expliquer doivent toutes être quantitatives. Si ce n'est pas le cas, transformer ce tableau en matrice de distance (voir fiche **101**) et réaliser le test sur les distances.

---

104. L'analyse de redondance (*RDA*)
105. L'analyse discriminante linéaire (*LDA*)
106. La régression PLS discriminante (*PLS-DA*)
107. L'analyse canonique des correspondances (*CCA*)
108. L'analyse des correspondances discriminante (*DCA*)
109. Analyser un ensemble de variables quantitatives

☛ Analyses sur une matrice de distance

Analyses asymétriques sur une matrice de distance

Le choix de l'analyse dépend de l'objectif :
1. L'objectif est de réaliser une ordination → Analyse de redondance sur matrice de distance
2. L'objectif est simplement de réaliser un test statistique → Analyser une matrice de distance.

---

110. L'analyse de redondance sur matrice de distance (*db-RDA*)
111. Analyser une matrice de distance

Analyses symétriques

Analyses symétriques sur deux tableaux

Les analyses suivantes peuvent porter sur des tableaux de variables et/ou sur les axes d'une précédente ordination. On peut ainsi coupler deux matrices de distance, un tableau de variables et une matrice de distance, un tableau de variables quantitatives et un tableau de variables qualitatives... Les possibilités sont nombreuses.
Le choix de l'analyse dépend surtout de l'habitude et de la tradition dans les différentes disciplines scientifiques. De façon générale :
- L'analyse PLS à deux blocs est employée surtout pour analyser la concordance entre deux tableaux de variables (obligatoirement quantitatives), sans contrainte sur leur nombre
- L'analyse procustéenne est employée surtout pour analyer la concordance entre deux tableaux de deux variables chacun (obligatoirement quantitatives) ou pour tester la concordance entre deux ordinations à deux axes chacune
- L'analyse de co-inertie est employée surtout pour analyser la concordance entre deux tableaux de variables (de tous types) ou pour tester la concordance entre deux ordinations, sans contrainte sur le nombre de variables/axes
- L'analyse de co-inertie procustéenne généralise l'analyse procustéenne au cas où au moins un tableau contient plus de deux variables. Dans le cas de deux tableaux de variables quantitatives, elle est plus facile à interpréter que l'analyse PLS à deux blocs et l'analyse de co-inertie.

---

112. L'analyse PLS à deux blocs (*2B-PLS*)
113. L'analyse procustéenne
114. L'analyse de co-inertie (*CIA*)
115. L'analyse de de co-inertie procustéenne (*PCIA*)

ANALYSES PORTANT SUR 2 OU PLUS DE 2 TABLEAUX

Analyses symétriques sur deux ou plus de deux tableaux

Le choix de l'analyse dépend de l'objectif :

1. L'objectif est d'identifier l'information commune à plusieurs tableaux → Analyse canonique des corrélations régularisée généralisée
2. L'objectif est de tester la concordance entre plusieurs tableaux et d'identifier une *configuration consensus, i.e.* une ordination « moyenne » (unique) de ces tableaux → Analyse procustéenne généralisée

116. L'analyse canonique des corrélations régularisée généralisée (*RGCCA*)
117. L'analyse procustéenne généralisée (*GPA*)

30. Graphiques de dispersion

Ce type de graphique représente toutes les données individuelles d'un vecteur, d'une matrice ou d'un tableau. Il permet d'avoir un aperçu de la variabilité des données et d'identifier les observations aberrantes. Son tracé est basé sur la fonction `stripchart()`.

Pour représenter un vecteur : `stripchart(vecteur)`.

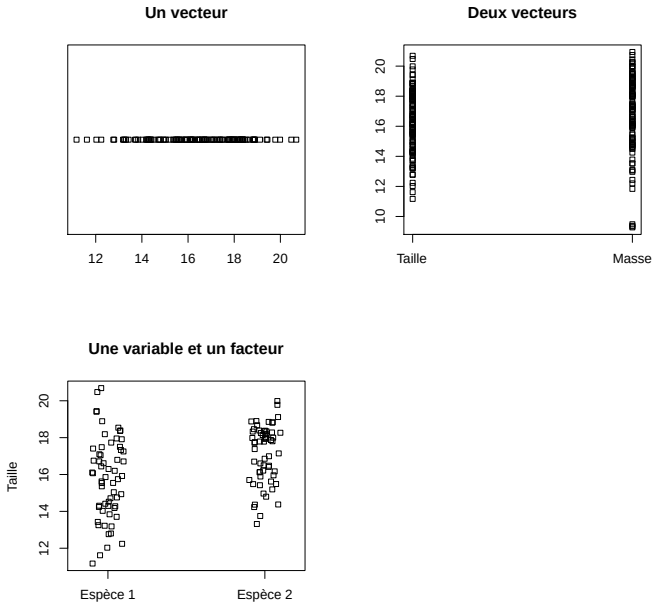
Pour représenter plusieurs vecteurs : `stripchart(list(vecteur1,vecteur2,...))`.
Pour donner un nom aux vecteurs sur le graphe, ajouter l'argument `group.names=c("Nom1","Nom2",...)`.

Pour représenter des données en fonction d'un facteur : `stripchart(reponse~facteur)` où les deux objets sont des vecteurs contenant la valeur de chaque individu (dans le même ordre). Le symbole `~` signifie « expliqué par » ou « en fonction de ».

Pour représenter les données verticalement, ajouter l'argument `vertical=TRUE`.

Pour que les valeurs identiques ne se superposent pas, ajouter l'argument `method="jitter"` (par défaut `method="overplot"`).

Pour ajouter un titre au graphe, utiliser l'argument `main="Titre"`.
Pour modifier la légende de l'axe horizontal, utiliser l'argument `xlab="Légende"`.
Pour modifier la légende de l'axe vertical, utiliser l'argument `ylab="Légende"`.
Pour (beaucoup) plus d'options graphiques, voir `?par`.



31. Histogrammes

— PACKAGE(S) ADDITIONNEL(S) UTILISÉ(S) DANS CETTE FICHE —

[1]RVAideMemoire

Ce type de graphique divise les données contenues dans un vecteur en classes, et représente chaque classe en effectif ou densité. Il permet d'avoir un aperçu de la distribution des données. Son tracé est basé sur la fonction `hist()`.

Pour représenter les classes en effectif : `hist(vecteur)`.
Pour représenter les classes en densité : `hist(vecteur,freq=FALSE)` (`freq=TRUE` par défaut, ce qui représente les effectifs).

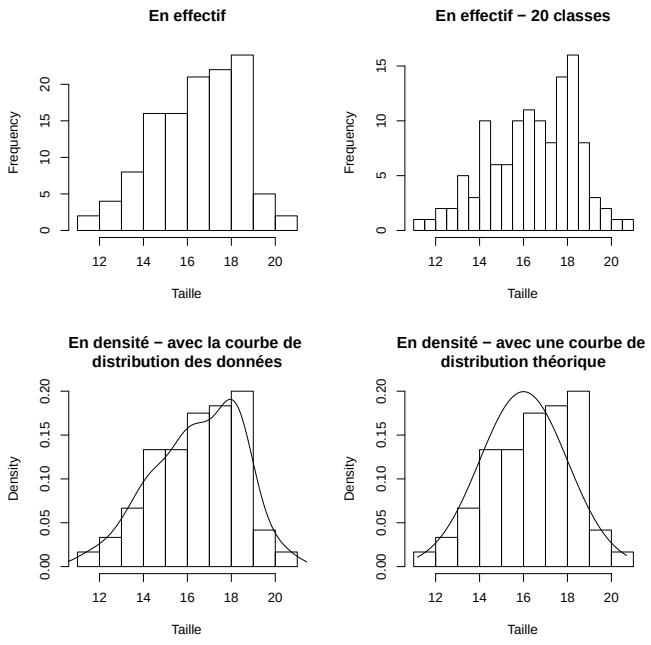
Pour ajouter une courbe de densité : `lines(density(vecteur))`. Une telle courbe ne peut être ajoutée que sur un histogramme tracé en densité.

Pour ajouter une courbe de distribution théorique : `lines(seq2(vecteur)`[1]`,dloi(seq2(vecteur)`[1]`,par))` où `loi` est la loi de probabilité choisie et `par` ses paramètres séparés par une virgule (voir fiches **19** à **28**).

Pour modifier le nombre de classes, ajouter l'argument `breaks=n` où `n` est le nombre de coupures souhaitées (il y a donc `n` + 1 classes).

Pour ajouter un titre au graphe, utiliser l'argument `main="Titre"`.
Pour modifier la légende de l'axe horizontal, utiliser l'argument `xlab="Légende"`.
Pour modifier la légende de l'axe vertical, utiliser l'argument `ylab="Légende"`.
Pour (beaucoup) plus d'options graphiques, voir `?par`.

Pour tracer l'histogramme d'une variable par niveau d'un facteur : `byf.hist(reponse~facteur)`[1] où `reponse` et `facteur` sont des vecteurs contenant la valeur de chaque individu pour les deux variables (dans le même ordre). Le symbole `~` signifie « expliqué par » ou « en fonction de ».



32. Boîtes à moustaches

Ce type de graphique représente de façon simplifiée la dispersion des données contenues dans un vecteur. Il permet d'avoir un aperçu de la distribution et de la variabilité des données, et d'identifier les observations aberrantes. Son tracé est basé sur la fonction `boxplot()`.

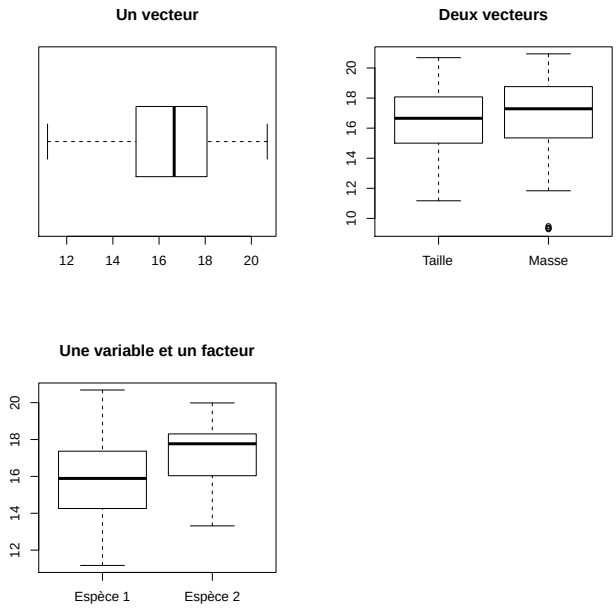
Pour représenter un vecteur : `boxplot(vecteur)`. Le trait épais représente la médiane, la boîte est formée par les valeurs des 1er et 3ème quartiles, et les moustaches mesurent au maximum 1,5 fois la longueur de l'interquartile (*i.e.* la différence 3ème quartile - 1er quartile). Les valeurs au-delà des moustaches sont représentées individuellement.

Pour représenter plusieurs vecteurs : `boxplot(list(vecteur1,vecteur2,...))`.
Pour donner un nom aux boîtes, ajouter l'argument `names=c("Nom1","Nom2",...)`.

Pour représenter une boîte par niveau d'un facteur : `boxplot(reponse~facteur)` où `reponse` et `facteur` sont des vecteurs contenant la valeur de chaque individu pour les deux variables (dans le même ordre). Le symbole `~` signifie « expliqué par » ou « en fonction de ».

Pour représenter les boîtes horizontalement, ajouter l'argument `horizontal=TRUE`.

Pour ajouter un titre au graphe, utiliser l'argument `main="Titre"`.
Pour modifier la légende de l'axe horizontal (si les boîtes sont tracées horizontalement), utiliser l'argument `xlab="Légende"`.
Pour modifier la légende de l'axe vertical (si les boîtes sont tracées verticalement), utiliser l'argument `ylab="Légende"`.
Pour (beaucoup) plus d'options graphiques, voir `?par`.



33. Graphiques en haricots

— PACKAGE(S) ADDITIONNEL(S) UTILISÉ(S) DANS CETTE FICHE —

[1]`beanplot`

Ce type de graphique combine les avantages d'un graphique de dispersion (*i.e.* représenter les données individuellement, voir fiche **30**), d'un histogramme (*i.e.* représenter la distribution des données, voir fiche **31**) et de boîtes à moustache (*i.e.* donner une vision globale de la distribution par niveau d'un facteur, voir fiche **32**).

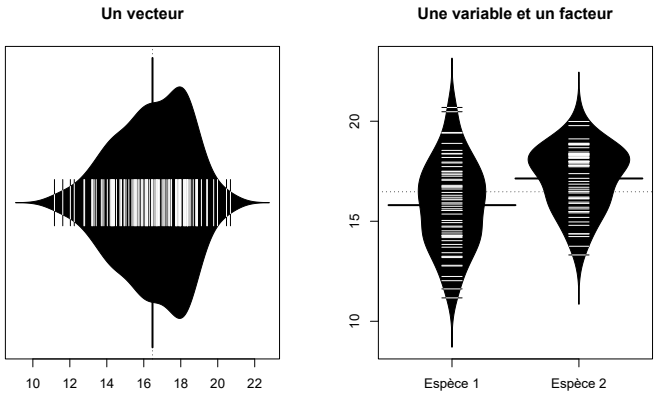
Pour représenter un vecteur : `beanplot(vecteur)`[1]. Chaque donnée individuelle est représentée par un petit segment horizontal (si deux données sont similaires le segment est deux fois plus long), la distribution est représentée verticalement en miroir, la moyenne est indiquée par un long segment épais et la moyenne générale par une droite en pointillés.

Pour représenter plusieurs vecteurs : `beanplot(list(vecteur1,vecteur2,...))`[1].
Pour donner un nom aux boîtes, ajouter l'argument `names=c("Nom1","Nom2",...)`.

Pour représenter une boîte par niveau d'un facteur : `beanplot(reponse~facteur)`[1] où `reponse` et `facteur` sont des vecteurs contenant la valeur de chaque individu pour les deux variables (dans le même ordre). Le symbole `~` signifie « expliqué par » ou « en fonction de ».

Pour représenter les boîtes horizontalement, ajouter l'argument `horizontal=TRUE`.

Pour ajouter un titre au graphe, utiliser l'argument `main="Titre"`.
Pour modifier la légende de l'axe horizontal (si les boîtes sont tracées horizontalement), utiliser l'argument `xlab="Légende"`.
Pour modifier la légende de l'axe vertical (si les boîtes sont tracées verticalement), utiliser l'argument `ylab="Légende"`.
Pour (beaucoup) plus d'options graphiques, voir `?beanplot` (notamment les arguments `what` et `col`) et `?par`.



34. Diagrammes en barres avec barres d'erreur

L'exemple présenté ici traite de moyennes et de barres d'erreur représentant des erreurs standards. Il peut bien sûr être adapté à n'importe quelles valeurs.

Un facteur

L'étape préliminaire est de rassembler les moyennes (une par modalité du facteur) dans un vecteur `moyennes` (contenant les valeurs dans l'ordre du graphe, de gauche à droite) et les erreurs standards (voir fiche **37**) dans un vecteur `erreurs` (avec les valeurs dans le même ordre que les moyennes). La fonction `tapply()` est très utile dans cette situation (voir fiche **35**).
La procédure est ensuite la suivante :

```
> abscisses <- barplot(moyennes)
> arrows(abscisses,moyennes-erreurs,abscisses,moyennes+erreurs,code=3,angle=90,length=0.15)
```

Deux facteurs

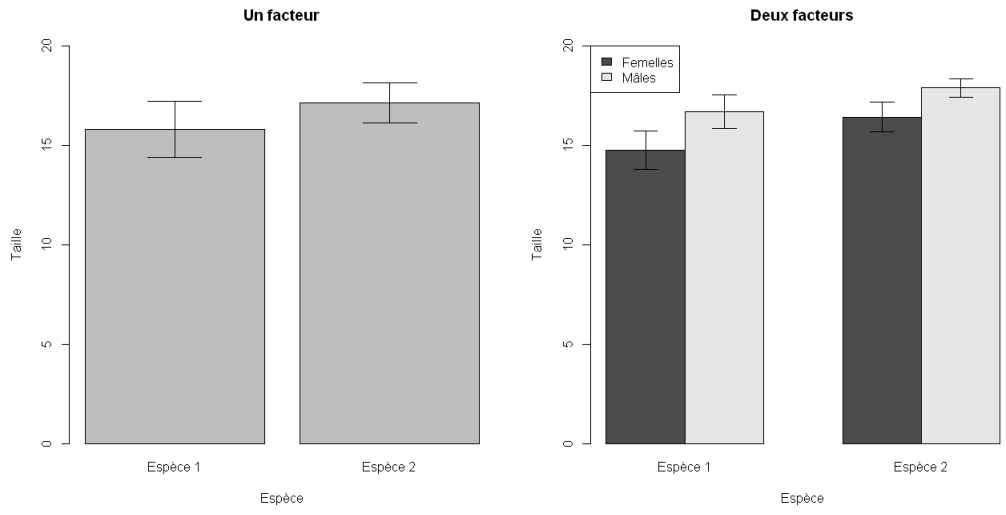
Moyennes et erreurs standards doivent être contenues dans des matrices. Ces matrices doivent avoir en lignes les modalités du 2nd facteur et en colonnes celles du 1er facteur (la fonction `tapply()` est là encore très utile, voir fiche **35**). La procédure est ensuite identique, il faut seulement ajouter l'argument `beside=TRUE` à la fonction `barplot()`.

L'argument `names.arg=noms` de la fonction `barplot()` ajoute ou modifie le nom des barres (`noms` étant un vecteur contenant les noms de gauche à droite), tandis que `legend=TRUE` ajoute une légende dans le cas de deux facteurs.

Pour ajouter un titre au graphe, utiliser l'argument `main="Titre"`.

Pour modifier la légende de l'axe vertical, utiliser l'argument `ylab="Légende"`.

Pour (beaucoup) plus d'options graphiques, voir `?barplot` (options spécifiques aux diagrammes en barre), `?legend` (options spécifiques de la légende) et `?par` (options graphiques générales).



35. Paramètres de position

— PACKAGE(S) ADDITIONNEL(S) UTILISÉ(S) DANS CETTE FICHE —

[1]RVAideMemoire

Les paramètres de position permettent de donner l'ordre de grandeur d'une série de données. Les trois plus fréquents sont :

— la moyenne : `mean(vecteur)` où `vecteur` est un vecteur contenant la valeur de chaque individu
— la médiane : `median(vecteur)`
— le mode : `mod(vecteur)`[1].

Si les vecteurs contiennent des données manquantes (`NA`), ajouter l'argument `na.rm=TRUE` aux fonctions `mean()` et `median()`. La fonction `mod()`[1] gère par défaut les données manquantes.

Pour calculer la valeur d'un paramètre par niveau d'un facteur, utiliser `tapply(vecteur,facteur, function(x) fonction)` où `vecteur` et `facteur` sont des vecteurs contenant la valeur de chaque individu pour les deux variables (dans le même ordre), et `fonction` la fonction à utiliser. Dans cette fonction, `vecteur` doit être remplacé par `x` : `mean(x)`, `median(x,na.rm=TRUE)`...

— EXEMPLE(S) —

```
> variable <- c(1:60)
> facteur <- factor(rep(LETTERS[1:3],each=20))
> tapply(variable,facteur,function(x) mean(x))
   A    B    C
10.5 30.5 50.5
```

Avec un deuxième facteur, la fonction renvoie une matrice :

```
> facteur2 <- factor(rep(letters[1:2],30))
> tapply(variable,list(facteur2,facteur),function(x) mean(x))
   A  B  C
a 10 30 50
b 11 31 51
```

Le premier facteur défini les *lignes* de la matrice, le second en définit les *colonnes*.

36. Paramètres de dispersion

— PACKAGE(S) ADDITIONNEL(S) UTILISÉ(S) DANS CETTE FICHE —

[1]RVAideMemoire

Les paramètres de dispersion permettent d'estimer la variabilité d'une série de données. Les trois plus fréquents sont :

- la variance : `var(vecteur)` où `vecteur` est un vecteur contenant la valeur de chaque individu
- l'écart-type (*standard deviation*) : `sd(vecteur)`
- le coefficient de variation : `cv(vecteur)`[1]. Le coefficient est par défaut exprimé en valeur absolue et en pourcentage.

Si les vecteurs contiennent des données manquantes (`NA`), ajouter l'argument `na.rm=TRUE` aux fonctions `var()` et `sd()`. La fonction `cv()`[1] gère par défaut les données manquantes.

Les fonctions `var()` et `sd()` calculent la variance et l'écart-type *non biaisés* (*i.e.* sur la base de $n-1$ et non n, n étant l'effectif de l'échantillon). La fonction `cv()`[1] est basée sur l'écart-type non biaisé.

Pour calculer la valeur d'un paramètre par niveau d'un facteur, utiliser `tapply(vecteur,facteur, function(x) fonction)` où `vecteur` et `facteur` sont des vecteurs contenant la valeur de chaque individu pour les deux variables (dans le même ordre), et `fonction` la fonction à utiliser. Dans cette fonction, `vecteur` doit être remplacé par `x` : `var(x)`, `sd(x,na.rm=TRUE)`...

— EXEMPLE(S) —

```
> variable <- c(1:60)
> facteur <- factor(rep(LETTERS[1:3],each=20))
> tapply(variable,facteur,function(x) sd(x))
   A    B    C
5.92 5.92 5.92
```

Avec un deuxième facteur, la fonction renvoie une matrice :

```
> facteur2 <- factor(rep(letters[1:2],30))
> tapply(variable,list(facteur2,facteur),function(x) sd(x))
     A    B    C
a 6.06 6.06 6.06
b 6.06 6.06 6.06
```

Le premier facteur défini les *lignes* de la matrice, le second en définit les *colonnes*.

Attention à ne pas confondre les paramètres de dispersion avec l'intervalle de confiance ou l'erreur standard (voir fiche **37**). Les paramètres de dispersion donnent une indication de la *variabilité des données*, tandis que l'intervalle de confiance et l'erreur standard donnent une indication de la *précision d'un paramètre de position* (voir fiche **35**). En particulier, la taille d'une série de données n'a pas d'impact sur sa dispersion (la variabilité n'augmente ou ne diminue pas avec le nombre d'individus) mais en a un sur la précision des paramètres de position (cette précision augmente avec le nombre d'individus).

37. Intervalle de confiance et erreur standard

— PACKAGE(S) ADDITIONNEL(S) UTILISÉ(S) DANS CETTE FICHE —

[1]`RVAideMemoire`

L'intervalle de confiance et l'erreur standard permettent d'estimer la précision d'une grande variété de paramètres. Les trois cas les plus courants sont une moyenne, une médiane (pour laquelle seul l'intervalle de confiance a du sens) et une proportion.

Attention à ne pas confondre l'intervalle de confiance ou l'erreur standard avec les paramètres de dispersion d'une série de données (voir fiche **36**). Les paramètres de dispersion donnent une indication de la *variabilité des données*, tandis que l'intervalle de confiance et l'erreur standard donnent une indication de la *précision d'un paramètre de position* (voir fiche **35**). En particulier, la taille d'une série de données n'a pas d'impact sur sa dispersion (la variabilité n'augmente ou ne diminue pas avec le nombre d'individus) mais en a un sur la précision des paramètres de position (cette précision augmente avec le nombre d'individus).

Pour toutes les fonctions calculant un intervalle de confiance, la précision de cet intervalle peut être modifiée grâce à l'argument `conf.level` (par défaut `conf.level=0.95`, ce qui calcule l'intervalle de confiance à 95 %).

Moyenne

Intervalle de confiance

Si l'effectif de la série de données est grand (≥ 30 individus), l'intervalle de confiance est calculé de façon paramétrique : `t.test(serie)$conf.int` où `serie` est un vecteur contenant la série de données.

Si l'effectif est petit (< 30 individus), l'intervalle de confiance est calculé de façon non paramétrique par bootstrap : `bootstrap(serie,function(x,i) mean(x[i]))`[1].

Erreur standard

Quel que soit l'effectif : `se(serie)`[1]. Voir fiche **36** pour calculer l'erreur standard par niveau d'un facteur.

Médiane

Intervalle de confiance

Si la série de données ne contient pas de 0 et quel que soit l'effectif : `wilcox.signtest(serie)$conf.int`[1].

Si la série de données contient au moins un 0 et quel que soit l'effectif : `wilcox.signtest(serie,mu=valeur)$conf.int`[1] où `valeur` est une valeur quelconque *absente* de la série de données.

Proportion

Intervalle de confiance

Quel que soit l'effectif : `binom.test(a,b)$conf.int` où `a` est le nombre d'individus de la catégorie d'intérêt et `b` l'effectif total.

— EXEMPLE(S) —

L'intervalle de confiance d'un sex-ratio de 9 femelles sur 25 individus (*i.e.* 0.36) est :

```
> binom.test(9,25)$conf.int
[1] 0.1797168 0.5747937
```

Erreur standard

Quel que soit l'effectif : `se(a,b)`[1].

38. Identification et suppression des données aberrantes

L'identification des données aberrantes est une étape obligatoire de toute analyse statistique. Elle se fait essentiellement *visuellement*, grâce à des graphes du type histogramme (voir fiche **31**) ou boîtes à moustaches (voir fiche **32**).

La suppression d'éventuelles données aberrantes est un point hautement plus délicat que leur simple identification. Supprimer une ou plusieurs donnée(s) aura nécessairement un effet sur les analyses qui vont suivre, et cet effet sera d'autant plus important que l'effectif de l'échantillon est faible. Il est donc tentant de supprimer les individus qui orientent les résultats vers des conclusions inverses à celles qui sont attendues.

Il n'y a globalement que deux raisons qui doivent pousser à éliminer une donnée :

— s'il y a manifestement eu une erreur technique dans la mesure ou dans la retranscription de la donnée (par exemple si l'appareil de mesure est défectueux)
— si la valeur de la mesure obtenue est biologiquement improbable pour la variable mesurée.

En dehors de ces deux cas, il y a de grandes chances pour que la valeur « aberrante » soit simplement une singularité biologique de l'individu mesuré. Cette singularité est l'expression de la variabilité naturelle de tout caractère biologique, rien ne justifie donc que l'individu soit supprimé de l'étude.

Dans tous les cas, l'identification et la suppression éventuelle de données aberrantes doit se faire *avant tout autre analyse*.

39. Démarche d'utilisation des modèles

L'utilisation des modèles en statistique (en tout cas en statistique appliquée à la biologie) suit une démarche qui est globalement toujours la même :

1. **Construction du modèle :** (i) on choisit le type de modèle que l'on utilise (linéaire, linéaire généralisé, non linéaire...) ; (ii) on choisit la loi de distribution sur laquelle le modèle doit être basé *a priori* ; (iii) on définit les relations entre les variables explicatives dans ce modèle. Dans le cadre de cet aide-mémoire, les étapes (i) et (ii) sont toujours proposées, tandis que l'étape (iii) doit nécessairement être réalisée par l'utilisateur puisqu'elle dépend de la question de recherche. Cette étape (iii) est basée sur la construction d'une *formule* (voir fiche **40**).
2. **Vérification de l'ajustement du modèle aux données :** il est important de faire cette vérification, car toute la suite de la démarche sera biaisée (voire complètement faussée) si le modèle n'est pas bien représentatif des données réelles. En pratique cette vérification passe par un examen des *résidus* du modèle (voir fiche **41**). Si le modèle n'est pas bien ajusté, il y a plusieurs possibilités : changer complètement de modèle, changer la loi sur laquelle il est basé, modifier les relations entre les variables explicatives, transformer les données... Dans tous les cas, tant que le modèle n'est pas bien ajusté aux données l'analyse ne doit pas aller plus loin.

La suite de la démarche peut prendre deux directions différentes, selon le type d'étude et le contexte :

— Approche par test de l'effet des variables explicatives :
   3. **Application d'un test sur le modèle :** l'effet de chaque terme de la formule est testé, ce qui conduit à l'obtention d'une *p-value* (voir fiche **42**).
   4. **Comparaisons multiples (si nécessaire) :** si un facteur à plus de deux niveaux a un effet significatif, réaliser des comparaisons multiples permet de conclure sur ceux qui diffèrent réellement (voir fiche **43**).

— Approche par sélection de modèle :
   3. **Sélection :** des modèles dérivés du premier mais n'incluant que certains termes sont comparés entre eux et avec le modèle complet. L'objectif est de sélectionner celui qui est le meilleur compromis entre l'ajustement aux données (qui augmente avec le nombre de variables explicatives) et la fiabilité du modèle (qui diminue quand ce nombre augmente) (voir fiche **44**).

Comment choisir entre approche par test et approche par sélection de modèle ? Il n'y a pas de règle absolue, tout dépend du jeu de données, des objectifs de l'étude et des traditions dans sa discipline scientifique. Si l'on veut généraliser, on peut dire globalement que : lorsque l'on travaille dans un cadre contrôlé, avec un petit nombre de variables explicatives, on choisit plutôt une approche par test ; lorsque l'on travaille dans un cadre peu ou pas contrôlé, souvent caractérisé par un nombre plus important de variables explicatives, on choisit plutôt une approche par sélection de modèle. L'approche par sélection est également privilégiée quand l'objectif est de retenir les variables explicatives les plus pertinentes pour faire de la prédiction sur de futures données (à partir du modèle sélectionné).

Techniquement parlant, l'approche par test travaille sur une seule hypothèse (l'hyothèse nulle H_0, voir fiche **14**), quand l'approche par sélection confronte plusieurs hypothèses et donne un poids relatif à chacune dans l'explication des données observées. Si l'on maîtrise ces concepts, cela peut aider à choisir l'approche la plus pertinente. Il est vrai également qu'en biologie, l'approche par test est certainement plus facile à appréhender. Non pas qu'elle soit réellement plus aisée à manipuler, mais c'est celle qui est massivement enseignée. Elle est donc plus familière.

Enfin, il y a deux points à toujours garder en tête :

— Quelle que soit l'approche choisie, il faut rester raisonnable sur le nombre de variables explicatives. En clair : qu'elles soient bien moins nombreuses que le nombre d'individus (pour donner un ordre de grandeur, on peut dire 10 fois moins nombreuses). Si le nombre d'individus de l'étude est très contraint, il faut donc être lucide sur le nombre de variables explicatives que l'on peut intégrer tout en assurant une analyse statistique fiable.

— On ne mélange pas l'approche par test et celle par sélection de modèle. Par exemple, tester l'effet des variables explicatives retenues après une procédure de sélection de modèle. Il a été montré que cela induit un biais qui surestime ces effets.

40. Construction de la formule d'un modèle

La construction de formules est un élément essentiel de l'analyse statistique. Elle peut être relativement simple si les notions suivantes sont bien comprises :
- variable à expliquer et variable explicative
- variable quantitative et facteur (voir fiche **11**)
- facteur fixe et facteur aléatoire (voir fiche **11**)
- plan d'échantillonnage (voir fiche **12**) et plan d'expérience (voir fiche **13**).

Le principe de toute formule est le suivant : `variables.à.expliquer~variables.explicatives` (le symbole `~` signifie « expliqué par » ou « en fonction de »). En pratique il n'y a le plus souvent qu'une seule variable à expliquer, ce qui simplifie la partie gauche de la formule. C'est dans la partie droite, correspondant aux variables explicatives, qu'il est nécessaire de définir les relations entre ces variables. Il est important de comprendre que ces relations *dépendent de la question de recherche*. Les définir ne se fait donc pas au hasard, elles doivent au contraire représenter pertinemment la situation.

Il existe trois types de relation entre deux variables explicatives, représentées par trois symboles différents :
- l'effet de chaque variable est analysé *séparément* : `A + B`.
- l'effet de chaque variable *ainsi que leur interaction* est analysé : `A * B`. Cette syntaxe est un raccourci strictement équivalent à `A + B + A:B`, qui signifie « variable A et variable B et l'interaction entre A et B ». Avec plus de variables, les choses se compliquent : `A * B * C` est équivalent à `A + B + C + A:B + A:C + B:C + A:B:C`, ce qui signifie « (i) les variables seules (A, B, C) et (ii) les interactions d'ordre 2 (entre A et B, entre A et C, entre B et C) et (iii) l'interaction d'ordre 3 (entre A, B et C). Il peut être tentant de mettre toutes les interactions possibles dans la formule, mais c'est une chose à éviter. En effet, à partir de l'ordre 3 les interactions sont très difficilement interprétables (ex : l'effet de A dépend de B, mais aussi de C...), sans compter que bien souvent les effectifs ne sont pas assez élevés pour évaluer des effets aussi fins. Il est toujours plus pertinent de se limiter aux effets que l'on peut expliquer.
- un cas particulier, celui de deux facteurs *emboîtés* (ou *hiérarchisés*) : `A/B`, qui signifie « variable A et variable B emboîtée dans A » (ex : un facteur population emboîté dans un facteur région). On peut très bien étendre la relation à `A/B/C` (ex : si on considère des facteurs pays, région et population).

Ces trois symboles (quatre en réalité avec le `:`) permettent de définir toutes les relations entre les variables explicatives, qu'elles soient quantitatives ou qualitatives.

La gestion des facteurs aléatoires (voir fiche **11** à **13**) est un peu spéciale. Elle fait appel à une syntaxe particulière, qui dépend de la fonction utilisée. On rencontre deux cas :
- tests simples (non paramétriques généralement) : le facteur aléatoire est noté en fin de formule après le symbole `|`. Par exemple, dans la formule `reponse~facteur|bloc` le facteur `facteur` est fixe et le facteur `bloc` est aléatoire.
- modèles mixtes : on appelle *mixte* un modèle qui contient au moins un facteur aléatoire. Plusieurs packages gèrent ces modèles (chacun avec une syntaxe différente), dans cet aide-mémoire c'est `lme4` qui est utilisé. Le facteur aléatoire est alors précisé, toujours en fin de formule, sous la forme `(1|facteur.alea)`. Par exemple, dans la formule `reponse~facteur+(1|bloc)` le facteur `facteur` est fixe et le facteur `bloc` est aléatoire. Plusieurs facteurs aléatoires peuvent être mis dans la formule, sous la forme `(1|facteur1) + (1|facteur2)` s'ils sont à effet indépendant ou `(1|facteur1/facteur2)` s'ils sont hiérarchisés. Il est possible de faire des choses bien plus complexes avec les modèles mixtes, voir **Bates** (2010) pour plus d'information.

Tous les fonctions créant un modèle, et plus généralement toutes les fonctions basées sur une formule, acceptent un argument `data`. Celui-ci permet de préciser le tableau de données dans lequel aller chercher les variables contenues dans la formule, ce qui évite d'allonger la syntaxe de la formule (et parfois de provoquer des erreurs).

41. Vérification de la validité d'un modèle

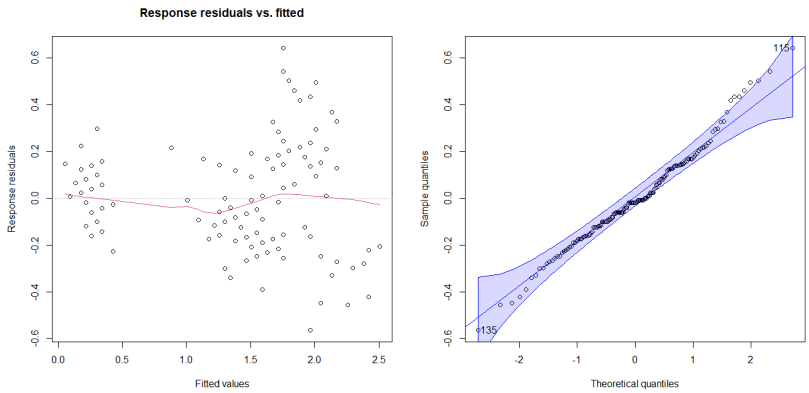
— PACKAGE(S) ADDITIONNEL(S) UTILISÉ(S) DANS CETTE FICHE —

[1]`RVAideMemoire`

Écrire un modèle pour analyser des données est une chose, mais il est très important de vérifier que ce modèle s'ajuste bien aux données. Si ce n'est pas le cas, toute analyse découlant de ce modèle serait *a minima* biaisée, voire totalement invalide.

Les vérifications à effectuer pour valider un modèle se font essentiellement en s'intéressant aux *résidus* de ce modèle (les résidus étant les écarts entre les valeurs réellement observées et celles prédites par le modèle, ces dernières étant nommées *fitted values*). Selon le type de modèle, on peut rencontrer trois conditions que les résidus doivent respecter (à vérifier dans cet ordre) :

1. Indépendance : cette condition est systématique et fondamentale. Si elle est respectée, cela veut dire que la relation sur laquelle est basée le modèle (linéaire le plus souvent) est bien respectée ; une violation flagrante de cette hypothèse doit absolument être corrigée. Le respect de cette condition se vérifie graphiquement grâce à la fonction `plotresid(modele)`[1] (sur le graphe de gauche si deux graphes sont tracés). Sur le graphe, l'hypothèse d'indépendance est acceptée lorsque l'orientation du nuage de points est horizontale (*i.e.* quand la ligne rouge – qui représente la courbe de tendance du nuage de points – ne s'éloigne pas trop de l'horizontale).
2. Variance : une condition sur la variance des résidus est fréquente. Plus précisément, la condition porte sur la relation entre variance des résidus et valeurs prédites par le modèle.
   Dans le cas le plus simple qui correspond aux modèles basées sur une loi normale (*i.e.* modèles linéaires (y compris mixtes), modèles non linéaires à équation connue…), l'hypothèse est que la variance des résidus est constante, indépendante des valeurs prédites (hypothèse d'*équivariance* ou d'*homoscédasticité*). Cela se vérifie sur le même graphe que la condition précédente : l'hypothèse d'équivariance est acceptée lorsque la dispersion verticale des points est à peu près constante sur toute la longueur de l'axe des abscisses.
   Dans le cas de modèles linéaires généralisés basés sur une loi autre que normale (et autres cas moins fréquents, en tout cas dans cet aide-mémoire), il y a une relation entre variance des résidus et valeurs prédites par le modèle. Si les résidus ont une variance plus grande que ce qui est théoriquement prévu on parle de *sur-dispersion* (fréquente), s'ils ont une variance plus petite on parle de *sous-dispersion* (rare). La dispersion théorique valant 1, on considère qu'il y a sur-dispersion au-dessus de 1,5 (seuil subjectif, certains diraient 2 voire plus), sous-dispersion au-dessous de 0,5. La façon de vérifier si la dispersion théorique est respectée est expliquée dans les fiches concernées.
3. Normalité : pour les modèles basés sur une loi normale, les résidus doivent suivrent une distrbution normale (dans les autres cas couverts par cet aide-mémoire, il n'y a pas de condition sur la distribution car celle-ci n'est pas unique). Cette condition se vérifie sur le second graphe renvoyé par la fonction `plotresid()`[1] (qui n'est tracé que si le modèle est basé sur une loi normale) : l'hypothèse de normalité est acceptée lorsque les points sont à peu près alignés sur une droite (voir fiche **65**).



42. Application d'un test sur un modèle

— PACKAGE(S) ADDITIONNEL(S) UTILISÉ(S) DANS CETTE FICHE —

[1]`car`

Après avoir vérifié que le modèle que l'on a construit est bien ajusté aux données (voir fiche **41**), l'effet des variables explicatives peut être testé. Il existe trois façons de réaliser ce test, que l'on appelle types I, II et III. S'il n'y a qu'une variable explicative, les trois types de test donnent exactement le même résultat. S'il y a plusieurs facteurs parmi les variables explicatives mais que les effectifs sont *équilibrés* (*i.e.* s'il y a autant d'individus dans toutes les classes d'un facteur et dans toutes les combinaisons des facteurs en interaction), il en est de même. Les trois types de test donnent des résultats différents s'il y a plusieurs facteurs et que les effectifs ne sont pas équilibrés. Il est important de comprendre d'où viennent ces différences, car elles reflètent des hypothèse statistiques (et donc biologiques) différentes. On prendra comme exemple un modèle du type `reponse~A+B+A:B`, où `A` et `B` sont des facteurs (voir fiche **40** pour une explication de la formule).

Type I : l'analyse est *séquentielle*, *i.e.* les termes sont testés dans l'ordre où ils apparaissent dans la formule (NB : lorsqu'un modèle est créé, **R** réorganise automatiquement la formule pour placer les facteurs seuls d'abord, puis les interactions d'ordre 2, les interactions d'ordre 3...). Dans notre exemple on a donc (i) un test de l'effet de `A` ; (ii) un test de l'effet de `B` *après avoir retiré* l'effet de `A` ; un test de l'effet de `A:B` après retiré les effets de `A` et `B`. Cela implique que les résultats du test pour les termes `A` et `B` sont *dépendants de l'ordre dans lequel ils apparaissent dans la formule*. Les hypothèses sous-jacentes sont rarement celles que l'on veut tester en biologie, aussi le type I est à oublier (excepté bien sûr si c'est exactement ce que l'on souhaite tester). Les tests de type I sont réalisés par la fonction `anova(modele)`. Attention, selon les modèles ce n'est pas le même test qui est appliqué.

Type II : l'analyse est *non séquentielle, i.e. indépendante de l'ordre des termes dans la formule*. Elle respecte par contre le *principe de marginalité*, qui stipule que « lorsqu'une interaction a un effet significatif, l'effet des variables de l'interaction, lorsqu'elles sont impliquées dans des termes d'ordre inférieur, est *marginal* devant celui de l'interaction ». Dit autrement, ce principe dit que si une interaction entre deux facteurs a un effet significatif, l'information est dans cette interaction et il est inutile de regarder l'effet des facteurs pris seuls (étendu à trois facteurs : si l'interaction entre les trois a un effet significatif, il est inutile de s'intéresser aux facteurs pris seuls et aux interactions d'ordre 2 impliquant ces facteurs). Concrètement, on considère donc que lorsque l'on teste l'effet d'un facteur, tous les termes d'ordre supérieur impliquant ce facteur n'ont pas d'effet. Dans notre exemple, cela donne (i) un test de l'effet de `A` après avoir retiré l'effet de `B` ; (ii) un test de l'effet de `B` après avoir retiré l'effet de `A` ; (iii) un test de l'effet de `A:B` après avoir retiré les effets de `A` et `B`. Cette approche est celle qui est systématiquement utilisée dans cet aide-mémoire, car dans la très grande majorité des cas c'est celle qui est la plus pertinente vis-à-vis des hypothèses biologiques. Les tests de type II sont réalisés par la fonction `Anova(modele,type="II")`[1] ou plus simplement `Anova(modele)`[1], le type II étant celui utilisé par défaut. Attention, selon les modèles ce n'est pas le même test qui est appliqué.

Type III : l'analyse est *non séquentielle* et *ne respecte pas* le principe de marginalité. Le test de l'effet d'un facteur est donc réalisé en prenant en compte les interactions impliquant ce facteur. Dans notre exemple, cela donne (i) un test de l'effet de `A` après avoir retiré les effets de `B` et `A:B` ; (ii) un test de l'effet de `B` après avoir retiré les effets de `A` et `A:B` ; (iii) un test de l'effet de `A:B` après avoir retiré les effets de `A` et `B`. Cette approche, même si elle peut paraître intéressante au premier abord, est très délicate à utiliser car elle revient à considérer qu'un facteur puisse ne pas avoir d'effet seul tout en modifiant l'effet d'un autre facteur (*i.e.* tout en ayant un effet ailleurs). Excepté si c'est réellement l'hypothèse que l'on veut tester, le type III est donc à oublier car il est rarement pertinent vis-à-vis des hypothèses biologiques. Les tests de type III sont réalisés par la fonction `Anova(modele,type="III")`[1]. Attention, selon les modèles ce n'est pas le même test qui est appliqué.

43. Comparaisons multiples basées sur un modèle

— PACKAGE(S) ADDITIONNEL(S) UTILISÉ(S) DANS CETTE FICHE —

[1]emmeans, [2]multcomp, [3]RVAideMemoire

Lorsqu'un test sur un modèle a révélé un effet significatif d'un facteur (ou d'une interaction impliquant au moins un facteur), il est nécessaire de réaliser des comparaisons multiples pour voir précisément quelles sont les modalités qui diffèrent. La méthode présentée ici est l'une des plus intéressantes car elle prend en compte l'effet des autres variables explicatives du modèle dans les comparaisons. En d'autres termes les comparaisons se font après avoir retiré la variation due aux autres variables explicatives. Ce ne sont donc plus les moyennes brutes (telles qu'on peut les calculer dans la fiche **35**) qui sont comparées, mais des moyennes *ajustées* en fonction des autres variables explicatives. La méthode en question est celle des *moyennes marginales estimées*, ou *Estimated Marginal Means* (abrégé le plus souvent en EMMeans).

Remarque 1 : dans le cas d'une interaction entre un facteur et une covariable, ce ne sont pas des moyennes qui sont comparées mais des *pentes* (les pentes de la relation entre la variable à expliquer et la covariable, calculées pour chaque modalités du facteur).

Remarque 2 : cette fiche concerne tous les modèles présentés dans ce document exceptés ceux analysant une réponse nominale à plus de deux catégories (voir fiche **51**) ou un décompte d'individus dans plus de deux catégories (voir fiche **63**).

Calcul et enregistrement des moyennes/pentes ajustées

Facteur seul ou interaction entre deux facteurs : moyennes

La première étape consiste à calculer les moyennes ajustées, qui sont stockées dans un objet appelé `EMM`. La syntaxe est du type : `EMM<-emmeans(modele,~facteur)`[1] où `facteur` est le nom du facteur ou de l'interaction d'intérêt (attention à ne pas oublier le symbole `~`). Pour calculer les moyennes séparément pour chaque niveau d'un autre facteur : `EMM<-emmeans(modele,~facteur1|facteur2)`[1] où `facteur1` est le facteur (ou l'interaction) dont on veut comparer les modalités, et `facteur2` le facteur pour lequel on veut réaliser les comparaisons à l'intérieur de chaque modalité.

Remarque 1 : si le modèle inclut au moins une covariable, les moyennes ajustées sont calculées *pour la valeur moyenne de la covariable*. On peut changer cela et calculer les moyennes ajustées pour n'importe quelle valeur de la covariable en ajoutant l'argument `at=list(covariable=valeur)`, où `covariable` est le nom de la covariable et `valeur` est la valeur de la covariable pour laquelle on veut calculer les moyennes ajustées. On peut jouer sur plusieurs covariables à la fois en les séparant par une virgule (à l'intérieur des parenthèses de `list()`).

Remarque 2 : pour les GLMMs basés sur une loi binomiale négative (créés avec la fonction `glmer.nb()` ; voir fiche **55**), il est nécessaire de préciser en plus *via* l'argument `data` le tableau de données sur lequel est basé le modèle. Dans tous les autres cas ce n'est normalement pas nécessaire, mais si jamais une erreur survient tester en ajoutant l'argument `data`.

Remarque 3 : pour les MLMs (voir fiche **109**), les moyennes ajustées peuvent être calculées séparément pour chaque variable à expliquer. Pour ce faire : `EMM<-emmeans(modele,~facteur|rep.meas)`[1]. Il est impératif d'utiliser la syntaxe `rep.meas`.

— EXEMPLE(S) —

On utilise un modèle appelé `modele` ayant pour formule `reponse~A*B+C` où `A`, `B` et `C` sont des facteurs (voir fiche **40** pour une explication de la formule).

Pour calculer la moyenne ajustée de chaque modalité de `A` :

```
> EMM <- emmeans(modele,~A)[1]
```

Pour les modalités de l'interaction `A:B` :

```
> EMM <- emmeans(modele,~A:B)[1]
```

Pour les modalités de `A` séparément pour chaque modalité de `C` :

```
> EMM <- emmeans(modele,~A|C)[1]
```

Pour les modalités de l'interaction `A:B` séparément pour chaque modalité de `C` :
`> EMM <- emmeans(modele,~A:B|C)`[1]

Si le modèle est un GLMM basé sur une loi binomiale négative et que le tableau de données s'appelle `tableau` :
`> EMM <- emmeans(modele,~A,data=tableau)`[1]
Si le modèle est un MLM et que l'on souhaite calculer les moyennes ajustées séparément pour chaque variable à expliquer :
`> EMM <- emmeans(modele,~A|rep.meas)`[1]

Interaction entre un facteur et une covariable : pentes

Les pentes se calculent *via* la fonction `emtrends()`[1], dont l'utilisation est très proche de `emmeans()`[1] : `pentes<-emtrends(modele,~facteur,var="covariable")`[1] où `facteur` peut être une interaction entre plusieurs facteurs. Attention aux guillemets autour du nom de la covariable.

Remarque 1 : la fonction `emtrends()`[1] n'est pas disponible pour les MLMs. Le cadre multivarié se prête en effet assez peu à l'analyse de l'effet de covariables (voir fiche **109**).

Remarque 2 : dans le cas d'une interaction entre un facteur et une covariable, il peut également être intéressant de comparer non des pentes, mais des moyennes ajustées (celles des modalités du facteur) pour une série de valeurs de la covariable. Une interaction significative indique en effet que la comparaison des moyennes doit donner des résultats différents selon la valeur de la covariable. Voir ci-dessus pour le calcul des moyennes pour une valeur donnée de la covariable.

EXEMPLE(S)

On utilise un modèle appelé `modele` ayant pour formule `reponse~F*V` où `F` est un facteur et `V` une covariable (voir fiche **40** pour une explication de la formule).

Pour calculer la pente de la relation entre `reponse` et `V` pour chaque modalité de `F` :
`> pentes <- emtrends(modele,~F,var="V")`[1]

Pour calculer la moyenne ajustée de `reponse` pour chaque modalité de `F` et pour une valeur de `V` égale à 1 :
`> EMM <- emmeans(modele,~F,at=list(V=1))`[1]
Même opération mais pour une valeur de `V` égale à 5 :
`> EMM <- emmeans(modele,~F,at=list(V=5))`[1]

Les matrices de contrastes

Les fonctions présentées dans cette fiche permettent de réaliser de manière simple toutes les comparaisons deux-à-deux possibles. Dans les cas où l'on veut réaliser des comparaisons personnalisées, il est nécessaire de commencer par créer une *matrice des contrastes*. Celle-ci est de la forme :

	Modalité1	Modalité2	Modalité3
Comparaison1	1	-1	0
Comparaison2	0	1	-1
Comparaison3	2	-1	-1

Dans cette matrice, les comparaisons (ou contrastes) sont représentées en lignes, tandis que les modalités du facteur (ou de l'interaction) sont en colonnes. Les conventions d'écriture des contrastes sont les suivantes :

— les modalités n'intervenant pas dans la comparaison doivent avoir une valeur nulle.
— les modalités à comparer doivent avoir une valeur non nulle et un signe opposé.
— la somme des valeurs positives et négatives d'un contraste doit être nulle.
— il est possible d'effectuer des regroupements de classes (comme dans la troisième ligne, où la modalité 1 est comparée au groupe formé par les modalités 2 et 3).

Attention, **R** considère les modalités du facteur dans l'ordre alphabétique, *i.e.* la 1ère colonne correspond à la 1ère modalité dans l'ordre alphabétique, et ainsi de suite.

La matrice des contrastes est concrètement un tableau, qui peut être créé directement dans **R** (voir fiche **5**) ou importé depuis un tableur (voir fiche **6**).

Réalisation des comparaisons

Pour réaliser toutes les comparaisons deux-à-deux : `cld(à.comparer,details=TRUE)`[2] où `à.comparer` est soit `EMM` soit `pentes`. La fonction `cld()`[2] renvoie à la fois la *p-value* de chaque comparaison et regroupe automatiquement les modalités dans des groupes de type a, ab, b… (nommés `1`, `12`, `2`…).

Dans le cas de comparaisons personnalisées, la procédure se fait en deux étapes :

```
> cont.emmc <- user.cont(contrastes)[3]
> contrast(à.comparer,"cont")[1]
```

où `contrastes` est la matrice des contrastes qui spécifie les comparaisons à réaliser.

Remarque 1 : pour pouvoir utiliser la fonction `cld()`[2] il est nécessaire d'avoir installé le package `multcompView`.

Remarque 2 : dans le cas d'un MLM (voir fiche **109**), les comparaisons multiples réalisées prennent en compte l'ensemble des variables à expliquer simultanément ; sauf si l'objet `EMM` a été créé en utilisant `rep.meas`, auquel cas les comparaisons sont réalisées séparément pour chaque variable à expliquer.

Tests de significativité des moyennes/pentes

Il est parfois intéressant de tester si les moyennes/pentes sont individuellement différentes de 0. La série de tests est réalisée automatiquement *via* `summary(à.comparer,infer=TRUE)`. Si jamais on veut tester si les moyennes/pentes sont individuellement différentes d'une autre valeur que 0, ajouter l'arguemnt `null=théo` où `théo` est la valeur théorique à laquelle les moyennes/pentes doivent être comparées.

Récupération des moyennes/pentes ajustées

Facteur seul ou interaction entre deux facteurs : moyennes

Représenter sur un diagramme en barres les moyennes ajustées par niveau d'un facteur (ou combinaison de deux facteurs) est souvent plus pertinent que de représenter les moyennes brutes. Cela permet d'illustrer la variation réellement due à ce facteur, après avoir retiré la variation due aux autres variables explicatives du modèle.

Pour tous les modèles sauf ceux analysant un rang dans un classement (voir fiche **52** pour la démarche à utiliser dans ce cas), récupérer les moyennes ajustées se fait de la même façon :

- Si la variable à expliquer n'a pas été transformée avant de créer le modèle : `valeurs<-as.data.frame(summary(EMM,type="response"))`. L'objet `valeurs` est un tableau dont la (les) première(s) colonne(s) correspond(ent) aux modalités du (des) facteur(s) précisé(s) dans `emmeans()`[1]. La colonne appelée `emmean`, `response`, `prob`, `rate` ou `hazard` contient les moyennes ajustées. La colonne `SE` contient l'erreur standard associée à chaque moyenne.
  Remarque : si le modèle est un GLM et qu'il est basé sur la fonction de lien logit empirique (voir fiches **48** et **60**), ajouter l'argument `tran=elogis()` à la fonction `summary()`.
- Si la variable à expliquer a été transformée avant de créer le modèle, ce qui en pratique ne concerne que certains LM(M)s et certaines régressions bêta :
  - LM(M) (voir fiche **76**) : `valeurs<-back.emmeans(EMM,transform="transfo",add=valeur)`[3] où `transfo` est le nom de la transformation entre guillemets (`"log"`, `"sqrt"`, `"4rt"` (*i.e.* racine quadratique : $\sqrt[4]{x}$ ou $x^{1/4}$), `"inverse"` ou `"logit"`) et `valeur` une éventuelle constante ajoutée à la variable à expliquer avant transformation. L'objet `valeurs` est un tableau dont la première colonne correspond aux modalités du (des) facteur(s) précisé(s) dans `emmeans()`[1]. La colonne `EMMean` donne les moyennes ajustées et les colonnes `SE.inf` et `SE.sup` donnent les bornes formées par l'erreur standard, dont l'intervalle est nécessairement asymétrique avec une transformation de la variable à expliquer.
  - Régression bêta (voir fiche **64**) : la procédure se fait en deux étapes :
    1. Recalculer les moyennes ajustées en ajoutant l'argument `mode="link"` à la fonction `emmeans()`[1].
    2. Utiliser `valeurs<-back.emmeans(EMM,transform="p.beta",n=nb)`[3] où `nb` est le nombre d'individus sur lequel la régression bêta a été construite (typiquement le nombre de lignes du jeu de données, mais attention à ne pas compter les individus pour lesquels il

y a des données manquantes car ces individus ne sont pas pris en compte par le modèle).

— Régression bêta mixte (voir fiche **64**) : `valeurs<-back.emmeans(EMM,transform="p.beta", n=nb)`[3].

EXEMPLE(S)

On utilise un modèle appelé `modele` dans lequel la variable à expliquer est transformée en $\sqrt{x}$, avec une formule `reponse~facteur` (voir fiche **40** pour une explication de la formule). Les moyennes ajustées sont récupérées de cette façon :

```
> EMM <- emmeans(modele,~facteur)[1]
> back.emmeans(EMM,transform="sqrt")[2]
```

Si la transformation est du type $ln(x+1)$:

```
> back.emmeans(EMM,transform="log",add=1)[2]
```

Si la transformation est du type $log_{10}(x+1)$:

```
> back.emmeans(EMM,transform="log",add=1,base=10)[2]
```

Une fois les moyennes et erreurs standards récupérées, elles peuvent être représentées sur un diagramme en barres (voir fiche **34**).

Remarque : dans le cas d'un MLM (voir fiche **109**), les moyennes ajustées n'ont de sens que si elles sont calculées pour chaque variable à expliquer. Il est donc impératif d'utiliser `rep.meas` pour les récupérer.

Interaction entre un facteur et une covariable : pentes

Les pentes sont récupérées *via* : `valeurs<-as.data.frame(summary(pentes))`. L'objet `valeurs` est un tableau dont la (les) première(s) colonne(s) correspond(ent) aux modalités du (des) facteur(s) précisé(s) dans `emtrends()`[1]. La colonne dont le nom se termine par `.trend` contient les pentes ajustées. La colonne `SE` contient l'erreur standard associée à chaque pente.

Remarque : en pratique les pentes ne sont directement interprétables que pour les LM(M)s.

44. Sélection de modèle

— PACKAGE(S) ADDITIONNEL(S) UTILISÉ(S) DANS CETTE FICHE —

[1]`MuMIn`, [2]`emmeans`

⚠ On ne mélange pas l'approche par test et celle par sélection de modèle (voir fiche **39**). ⚠

On utilise le plus souvent la démarche de sélection de modèle lorsque le nombre de variables explicatives est relativement important (typiquement quand on ne sait pas quelle variable peut avoir un effet, et que l'on en mesure donc toute une série), ou quand le modèle servira ensuite à faire de la prédiction. L'objectif est d'aboutir au meilleur compromis entre (i) un bon ajustement aux données (qui augmente avec le nombre de variables explicatives) et (ii) une estimation fiable des paramètres du modèles et des prédictions qu'il fait (qui diminue quand le nombre de variables explicatives augmente).

Choix du critère de sélection

La sélection est basée sur une valeur (appelée *critère*) représentant ce compromis. Plus elle est *faible*, meilleur est le compromis. Il existe plusieurs critères, les plus utilisés sont le Critère d'Information d'Akaike (AIC) et ses dérivés. Choisir le critère approprié dépend (i) du ratio entre le nombre d'individus (n) et le nombre de paramètres du modèle initial (df), et (ii) du fait que le modèle initial soit basé sur une loi de distribution explicite ou sur une loi « quasi » (quasi-Poisson, quasi-binomiale…) :

	Loi explicite (pas « quasi »)	**Loi « quasi »**
$\frac{n}{df} > 40$	AIC	QAIC
$\frac{n}{df} < 40$	AICc	QAICc

Pour se faire une idée du nombre de paramètres du modèle initial, on peut généralement appeler `summary(modele)` où `modele` est le nom du modèle. Le nombre de paramètres vaut *au minimum* le nombre du lignes du tableau souvent appelé `Coefficients`, `Fixed effects` ou `Parameters`. En cas de doute, on peut simplement se référer au nombre de variables explicatives : il y a toujours *plus* de paramètres que de variables explicatives. Donc s'il n'y a pas au moins 40 fois plus d'individus que de variables explicatives, le ratio $\frac{n}{df}$ est nécessairement inférieur à 40.

Remarque : ne jamais utiliser le coefficient de détermination R^2 comme critère de sélection, car celui-ci mène toujours à sélectionner un modèle contenant beaucoup de variables explicatives, sans tenir compte de la fiabilité des estimations.

Réalisation de la sélection

On peut réaliser la sélection de modèle automatiquement grâce à la fonction `dredge()`[1]. Cela se fait en deux étapes :

1. Créer le modèle initial (aussi appelé modèle complet car il contient toutes les variables explicatives et d'éventuelles interactions ; voir fiche **40** pour la syntaxe de la formule d'un modèle). Celui-ci doit obligatoirement inclure l'argument `na.action="na.pass"`.
   Remarque 1 : si le modèle initial est un Modèle Linéaire Mixte (LMM), il doit également contenir l'argument `REML=FALSE`.
   Remarque 2 : si le modèle initial est un Modèle Linéaire Généralisé (GLM) basé sur une loi quasi-binomiale (voir fiche **60**), remplacer l'argument `family="quasibinomial"` par `family="quasibinomial.QAIC"`. Le package `RVAideMemoire` doit être chargé pour que cela fonctionne.
   Remarque 3 : si le modèle initial est un GLM basé sur une loi quasi-Poisson (voir fiche **55**), remplacer l'argument `family="quasipoisson"` par `family="quasipoisson.QAIC"`. Le package `RVAideMemoire` doit être chargé pour que cela fonctionne.

2. Appliquer la procédure automatique : `selection<-dredge(modele,rank=critere)`[1] où `critere` est `"AIC"`, `"AICc"`, `"QAIC"`ou `"QAICc"` (entre guillemets).
   Remarque : dans le cas d'un modèle initial basé sur une loi « quasi », il est nécessaire d'ajouter l'argument `chat=valeur` où `valeur` est la valeur du paramètre de dispersion du modèle (obtenu *via* `summary(modele)`, typiquement après `Dispersion parameter for quasi[] family taken to be`).

EXEMPLE(S)

Avec un modèle du type :
```
> modele <- glm(formule,family="quasipoisson.QAIC",na.action="na.pass")
```
La valeur du paramètre de dispersion est récupérée *via* :
```
> summary(modele)
[...]
(Dispersion parameter for quasipoisson family taken to be 1.126135)
[...]
```
Et la sélection automatique peut ensuite être réalisée :
```
> selection <- dredge(modele,rank="QAICc",chat=1.126135)
```
[1]
La démarche est identique pour les modèles à loi `quasibinomial` et `quasi`.

Parmi les arguments facultatifs de la fonction `dredge()`[1], deux sont particulièrement intéressants :
— `m.max=valeur`, où `valeur` est le nombre maximal de variables explicatives à intégrer dans les modèles à tester
— `fixed=variables`, où `variables` est un vecteur contenant le nom des variables explicatives à intégrer dans tous les modèles à tester (entre guillemets).

Interprétation des résultats de la sélection

La fonction `dredge()`[1] renvoie un tableau dont chaque ligne correspond à un modèle obtenu à partir du modèle initial (en enlevant un ou plusieurs termes). Il y a une colonne par terme (variable explicative ou interaction) : si elle contient une valeur ou un symbole + le terme est inclus dans le modèle en question, si elle est vide le terme n'est pas inclus. Les modèles sont classés selon le critère choisi, du meilleur (en haut) au moins bon (en bas).

Remarque 1 : si le modèle initial est un modèle mixte (LMM ou GLMM), le (ou les) facteur(s) aléatoire(s) sont obligatoirement inclus dans tous les modèles générés.

Remarque 2 : il est normal qu'un des modèles ne contienne aucune variable explicative. On l'appelle le modèle nul.

Remarque 3 : la colonne `df` donne le nombre de paramètres de chaque modèle.

Un valeur importante est celle de la colonne `weight` (nommé *Akaike weight* ou w). Elle correspond à la *probabilité qu'un modèle particulier soit le meilleur* pour expliquer les données. Il ne faut donc pas conclure simplement en prenant le modèle en haut de la liste et en disant qu'il est le meilleur. Tout l'intérêt de l'approche par sélection de modèle est de ne pas trancher de façon aussi abrupte, mais de confronter les différents modèles et de leur donner une probabilité relative. Deux modèles qui ont des w très proches doivent être considérés comme équivalents, même si l'un a un w légèrement supérieur.

Remarque : on peut également calculer l'importance relative de chaque terme du modèle initial (variable explicative ou interaction) *via* `importance(selection)`[1].

Pour conclure sur le (ou les) meilleur(s) modèle(s), on utilise le plus souvent les valeurs de la colonne `delta` (nommé ΔAIC, ΔAICc, ΔQAIC ou ΔQAICc selon le critère utilisé), qui représente l'écart d'AIC/AICc/QAIC/QAICc entre un modèle particulier et le « meilleur » modèle (celui tout en haut de la liste). On considère classiquement que si aucun modèle n'a un $\Delta \leq 2$, alors le modèle avec le plus faible AIC/AICc/QAIC/QAICc est l'unique meilleur modèle. À l'inverse, si un ou plusieurs modèles ont un $\Delta \leq 2$, il n'y a pas un meilleur modèle unique mais un meilleur ensemble de modèles. Dans ce dernier cas, il est possible d'aller plus loin grâce à une procédure dite de *model averaging*.

Le *model averaging*

L'objectif d'une procédure de *model averaging* est de créer un modèle unique qui combine un ensemble de modèles différents. Donc d'obtenir *un seul* meilleur modèle à partir d'un meilleur *ensemble* de modèles. Chaque modèle de l'ensemble contribue au modèle synthétique à hauteur de son w (tous les modèles n'ont donc pas le même poids).

Pour réaliser la procédure : `mod.avg<-model.avg(selection,subset=(delta<=seuil),fit=TRUE)`[1] où `seuil` correspond au Δ maximal toléré (classiquement 2).

Remarque : dans le cas d'un modèle initial basé sur une loi « quasi », il est nécessaire d'ajouter l'argument `dispersion=valeur` où `valeur` est la valeur du paramètre de dispersion de ce modèle (la même que celle qui avait été fournie à la fonction `dredge()`[1]).

EXEMPLE(S)

À partir de l'exemple précédent, le *model averaging* est réalisé de cette manière :

```
> mod.avg <- model.avg(selection,subset=(delta<=2),dispersion=1.126135,fit=TRUE)[1]
```

La démarche est identique pour les modèles à loi `quasibinomial` et `quasi`.

L'interprétation des résultats du *model averaging* passe par :

— le calcul de l'importance de chaque terme inclus dans le modèle synthétique : `importance(mod.avg)`[1]

— pour une covariable : la valeur de son paramètre associé (*i.e.* le sens de la relation avec la variable à expliquer, qui dépend du signe du paramètre), dans la colonne `Estimate` du tableau `(full average)` renvoyé par `summary(mod.avg)` (ne pas tenir compte du tableau `(conditional average)`)

— pour un facteur fixe : la moyenne ajustée de chaque modalité dans le modèle synthétique, obtenue *via* `emmeans(mod.avg,~facteur)`[2] (voir fiche **43**).
Remarque : il n'est pas dans la logique de l'approche par sélection de modèle de réaliser des comparaisons deux-à-deux de ces moyennes.

On peut utiliser le modèle issu de la procédure de *model averaging* pour faire de la prédiction à partir de nouvelles données (à condition de fixer la valeur de *toutes* les variables explicatives). Pour cela, créer un tableau contenant une colonne par variable explicative (les noms de colonnes doivent être *strictement* identiques aux noms des variables du modèle), et remplir chaque ligne en fonction des valeurs pour lesquelles la prédiction doit être faite (il y a donc une prédiction par ligne de ce tableau). Puis : `predict(mod.avg,newdata=tableau)`.

Remarque 1 : si le modèle initial est un GLM(M), ajouter les arguments `type="link"` et `backtransform=TRUE`.

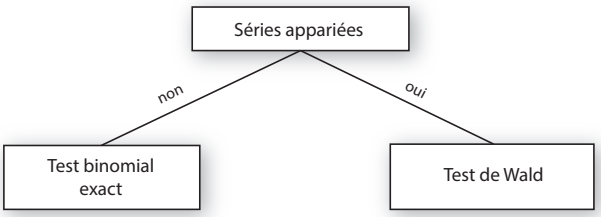
Remarque 2 : si le modèle initial est un modèle mixte (LMM ou GLMM), les valeurs du (ou des) facteur(s) aléatoire(s) peuvent être données ou non. Si ce n'est pas le cas, il est nécessaire d'ajouter l'argument `re.form=NA`. Les prédictions prennent alors en compte l'effet moyen de toutes les modalités du (ou des) facteur(s) aléatoire(s) du modèle.

45. Conformité d'une probabilité à une valeur théorique

— PACKAGE(S) ADDITIONNEL(S) UTILISÉ(S) DANS CETTE FICHE —

[1]`RVAideMemoire`

Pour choisir le test approprié :



Séries non appariées

Test binomial exact (non paramétrique)

Pour réaliser le test : `binom.test(n1,n,p=proba.theo)` où `n1` est le nombre d'individus de la catégorie d'intérêt, `n` l'effectif total et `proba.theo` la probabilité théorique de la catégorie d'intérêt (0.5 par défaut).

— EXEMPLE(S) —

On veut comparer le sex-ratio (ici la proportion de femelles) d'un échantillon de 20 individus contenant 7 femelles et 13 mâles à un sex-ratio équilibré (donc une probabilité de 0.5) :

```
> binom.test(7,20,p=0.5)
```

ou plus simplement puisque `p=0.5` par défaut :

```
> binom.test(7,20)
```

Séries appariées

Test de Wald (paramétrique)

Pour réaliser le test : `wald.ptheo.test(reponse,blocs,p=proba.theo)`[1] où `reponse` est la réponse de chaque individu (sous forme numérique ou d'un facteur, en tout cas binaire) et `blocs` un facteur (aléatoire) contenant le groupe de chaque individu (dans le même ordre que `reponse`). Si `reponse` est codée sous forme 0/1, la probabilité du groupe 1 est testée ; si `reponse` est un facteur, la probabilité de la 2nde modalité est testée.

46. Conformité de plusieurs probabilités à des valeurs théoriques – 2 classes

— PACKAGE(S) ADDITIONNEL(S) UTILISÉ(S) DANS CETTE FICHE —

[1]`RVAideMemoire`

Test du χ^2 de conformité (non paramétrique)

Conditions : les effectifs théoriques doivent tous être non nuls et 80 % d'entre eux doivent être ≥ 5 (« règle de Cochran », voir ci-dessous).

Les effectifs théoriques sont obtenus *via* `chisq.bin.exp(reponse~facteur,p=proba.theo)`[1] où `reponse` et `facteur` sont des vecteurs contenant la valeur de chaque individu pour les deux variables (dans le même ordre), et où `proba.theo` est un vecteur contenant la probabilité théorique dans chaque modalité (dans l'ordre des modalités de `facteur`). Si `reponse` est codée sous forme 0/1, la probabilité du groupe 1 est testée ; si `reponse` est un facteur, la probabilité de la 2nde modalité est testée. Le symbole `~` signifie « expliqué par » ou « en fonction de ». Les effectifs théoriques permettent de vérifier si la règle de Cochran est respectée.

Remarque : les probabilités théoriques sont *indépendantes* entre les modalités de `facteur`. Elles n'ont pas à donner une somme de 1, puisque les modalités ne sont pas comparées entre elles, mais chacune est testée pour sa propre probabilité théorique.

Pour réaliser le test : `chisq.theo.bintest(reponse~facteur,p=proba.theo)`[1].

Une *p-value* significative indique qu'au moins une probabilité diffère de sa valeur théorique, sans préciser la(les)quelle(s). Il est dans ce cas nécessaire de réaliser des comparaisons deux-à-deux pour identifier la (les) probabilité(s) en question, *via* `prop.bin.multcomp(reponse~facteur,p=proba.theo)`[1].

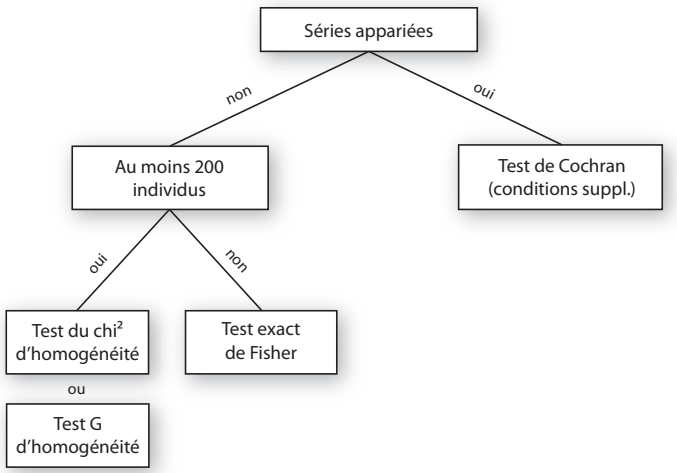
Il peut arriver que les comparaisons deux-à-deux n'indiquent aucune différence significative, contrairement au test global. Dans ce cas, la solution la plus prudente est de considérer qu'on ne peut pas savoir quelle modalité est responsable du rejet de l'hypothèse nulle dans le test global.

47. Comparaison de plusieurs probabilités – 2 classes

— PACKAGE(S) ADDITIONNEL(S) UTILISÉ(S) DANS CETTE FICHE —

`[1]RVAideMemoire`

Pour choisir le test approprié :



Le test exact de Fisher est toujours le plus fiable mais son temps de calcul augmente grandement avec le nombre d'individus. Lorsque l'effectif est suffisamment grand, l'approximation faite par les tests du χ^2 et G est assez satisfaisante pour qu'ils puissent être utilisés. Les résultats de ces deux tests sont très semblables ; choisir entre l'un et l'autre relève plus d'une habitude que d'une raison statistique.

Dans les tests suivants, si `reponse` est codée sous forme 0/1, la probabilité du groupe 1 est testée ; si `reponse` est un facteur, la probabilité de la 2nde modalité est testée.

Séries non appariées

Test exact de Fisher (non paramétrique)

Pour réaliser le test : `fisher.bintest(reponse~facteur)`[1] où `reponse` et `facteur` sont des vecteurs contenant la valeur de chaque individu pour les deux variables (dans le même ordre). Le symbole `~` signifie « expliqué par » ou « en fonction de ».

Si la *p-value* du test est significative, cela indique qu'au moins deux classes du facteur ont un effet différent sur la variable à expliquer (sans préciser lesquelles). La fonction réalise alors automatiquement toutes les comparaisons deux-à-deux possibles par une série de tests exacts de Fisher.

Il peut arriver que les comparaisons deux-à-deux n'indiquent aucune différence significative, contrairement au test global. Dans ce cas, la solution la plus prudente est de considérer qu'on ne peut pas savoir quelles probabilités sont responsables du rejet de l'hypothèse nulle dans le test global.

Test du χ^2 d'homogénéité (non paramétrique)

Pour réaliser le test : `chisq.bintest(reponse~facteur)`[1].

Si la *p-value* est significative, les comparaisons deux-à-deux sont réalisées automatiquement par une série de tests du χ^2 d'homogénéité.

Test G d'homogénéité (non paramétrique)

Pour réaliser le test : `G.bintest(reponse~facteur)`[1].

Si la *p-value* est significative, les comparaisons deux-à-deux sont réalisées automatiquement par une série de tests G d'homogénéité.

Séries appariées

Test Q de Cochran (non paramétrique)

Conditions : le plan d'expérience doit être en blocs aléatoires complets sans répétition (voir fiche **13**).

Pour réaliser le test : `cochran.qtest(reponse~fact.fixe|fact.alea)`[1] où `variable`, `fact.fixe` et `fact.alea` sont des vecteurs contenant la valeur de chaque individu (dans le même ordre) pour la variable à expliquer, le facteur (fixe) dont on veut comparer les modalités et le facteur (aléatoire) servant à définir les séries appariées, respectivement.

Si la *p-value* du test est significative, cela indique qu'au moins deux classes du facteur fixe ont un effet différent sur la variable à expliquer (sans préciser lesquelles). La fonction réalise alors automatiquement toutes les comparaisons deux-à-deux possibles par une série de tests des signes de Wilcoxon.

48. Analyser une réponse binaire

— PACKAGE(S) ADDITIONNEL(S) UTILISÉ(S) DANS CETTE FICHE —

[1]`lme4`, [2]`RVAideMemoire`, [3]`MuMIn`, [4]`car`

⚠ L'analyse présentée dans cette fiche est basée sur un modèle. Il est indispensable de maîtriser les notions des fiches **39** à **42**. ⚠

Réponse

Avec une réponse binaire (*i.e.* définissant deux catégories), l'une des deux catégories est définie comme « catégorie d'intérêt ». Ce qui est analysé est la *probabilité qu'un individu soit de cette catégorie d'intérêt*, appelée « probabilité de réponse » dans cette fiche.

La variable à expliquer peut être codée numériquement (sous forme 0/1) ou être un facteur à deux niveaux. Si elle est codée sous forme 0/1, la catégorie 1 est celle d'intérêt ; si elle est un facteur, la 2nde modalité est la catégorie d'intérêt.

Modèle utilisé

Si le principe de l'analyse est le même qu'il y ait des séries appariées ou non, quelques modalités diffèrent. La première est que sans séries appariées le modèle utilisé est un Modèle Linéaire Généralisé (*Generalized Linear Model* ou GLM), tandis qu'avec des séries appariées c'est un Modèle Linéaire Généralisé Mixte (*Generalized Linear Mixed Model* ou GLMM ; « mixte » sous-entendant « contenant au moins un facteur aléatoire », facteurs que l'on utilise précisément pour identifier les séries appariées (voir fiche **11**)).

Contrairement au Modèle Linéaire (et sa variante Mixte, voire fiche **76**), les GLM(M)s ne sont pas basés sur une loi normale. Dans le cas d'une réponse binaire, la loi à utiliser est une loi binomiale.

Remarque : il existe en fait plusieurs types de GLM(M)s basés sur une loi binomiale. Le cas décrit ici – le plus fréquent – est celui d'un modèle *logistique*. Lorsque toutes les variables explicatives sont des covariables, on est dans le cas particulier d'une *régression logistique*.

Construction du modèle

Pour créer le modèle :

— sans séries appariées : `modele<-glm(formule,family="binomial")`

— avec des séries appariées : `modele<-glmer(formule,family="binomial")`[1].

Voir fiche **40** pour une explication détaillée de la construction d'une formule. De manière générale, on peut dire que :

— inclure un facteur permet de tester si les probabilités de réponse diffèrent entre les niveaux de ce facteur.

— inclure une covariable permet de tester s'il existe une relation entre cette covariable et la variable à expliquer.

— inclure une interaction entre deux variables explicatives permet de tester si l'effet de l'une dépend de l'autre (le raisonnement est le même pour une interaction incluant plus de deux termes). Le cas particulier d'une interaction entre une covariable et un facteur permet de tester si la relation entre la covariable et la variable à expliquer est différente selon la modalité du facteur.

Remarque : dans le cas où au moins une des variables explicatives est un facteur, et que tous les individus d'au moins une modalité ont la même valeur (0 ou 1), l'ajustement du modèle ne peut pas se faire correctement. Dans ce cas, remplacer `family="binomial"` par `family=binomial(link=elogis()`[2]`)`. Trois avertissements sont alors renvoyés à la création du modèle, ne pas en tenir compte. Cette astuce n'est cependant applicable qu'aux GLMs, pas aux GLMMs. Si le modèle est un GLMM, on ne peut simplement pas intégrer proprement dans l'analyse la modalité pour laquelle toutes les valeurs de la variable à expliquer sont identiques.

Vérification de l'ajustement aux données

Avant d'aller plus loin, il est indispensable de vérifier que le modèle s'ajuste bien aux données. Cette étape est fondamentale, et ce pour tout modèle, car un test basé sur un modèle mal ajusté n'est simplement pas fiable (ni valide).

La seule condition à vérifier est l'indépendance des résidus du modèle avec les valeurs prédites (voir fiche **41** pour une explication détaillée). Si cette hypothèse n'est pas respectée de façon flagrante, changer la *fonction de lien* sur laquelle est basé le modèle. Par défaut cette fonction de lien est le logit et n'est pas précisée (`family="binomial"` est un raccourci pour `family=binomial(link="logit")`). Tester les fonctions de lien probit (`family=binomial(link="probit")`), cauchit (`family=binomial(link="cauchit")`), log (`family=binomial(link="log")`) ou *complementary log-log* (`family=binomial(link="cloglog")`), *i.e.* les quatre autres acceptées par la loi binomiale.

Remarque 1 : dans le cas de l'utilisation de la fonction `elogis()`[2], la fonction de lien est appelée « logit empirique ». Cette fonction de lien est difficile à remplacer par une autre. Si l'ajustement du modèle n'est pas bon avec cette fonction de lien, on ne peut simplement pas intégrer proprement dans l'analyse la modalité pour laquelle toutes les valeurs de la variable à expliquer sont identiques.

Remarque 2 : dans tous les cas où la fonction de lien n'est ni le logit ni le logit empirique, le modèle n'est plus un modèle logistique.

Capacité explicative globale

On peut estimer la capacité explicative globale d'un modèle grâce au *coefficient de détermination* (R^2), qui représente la proportion de la variance de la variable à expliquer qui est expliquée par les variables explicatives. Ce coefficient est obtenu *via* `r.squaredGLMM(modele)`[3] (ligne `delta`). La fonction renvoie en fait deux valeurs : le R^2 *marginal* (`R2m`) qui correspond à la part de la variance expliquée uniquement par les facteurs fixes et covariables, et le R^2 *conditionnel* (`R2c`) qui correspond à la part de la variance expliquée par l'ensemble des variables explicatives (fixes et aléatoires). Dans le cas d'un GLM les deux valeurs sont identiques puisqu'il n'y a pas de facteur aléatoire.

Test(s)

Quel que soit le modèle, l'effet des variables explicatives est testé par la même fonction : `Anova(modele)`[4] (voir fiche **42** pour une explication détaillée des hypothèses testées). Cependant, ce ne sont pas les mêmes tests qui sont réalisés selon le modèle :

— GLM : la fonction réalise un test du rapport des vraisemblances (*Likelihood Ratio Test* ou LR Test) – en fait un test par terme du modèle (*i.e.* un par ligne du tableau renvoyé).

— GLMM : la fonction réalise un test de Wald – en fait un test par terme du modèle (*i.e.* un par ligne du tableau renvoyé).

Si un facteur (ou une interaction impliquant un facteur) a un effet significatif, il est nécessaire de réaliser des comparaisons multiples pour identifier les modalités (ou combinaisons de modalités) qui diffèrent. Voir fiche **43** pour réaliser ces comparaisons.

Si une covariable a un effet significatif, la direction de son effet est donnée par le *signe* du paramètre associé. Les valeurs de tous les paramètres du modèle sont obtenues *via* `summary(modele)`. Elles sont appelées `Estimate` et se trouvent dans le tableau `Coefficients` pour un GLM, `Fixed effects` pour un GLMM. Si le coefficient portant le nom de la covariable est négatif, la probabilité de réponse diminue quand la valeur de la covariable augmente ; s'il est positif, la probabilité augmente quand la valeur de la covariable augmente.

Prédiction à partir du modèle

L'intérêt des modèles est à la fois de comprendre l'importance des différentes variables explicatives sur la variable à expliquer, mais également de prédire la valeur que prendrait cette variable à expliquer pour des valeurs *connues* des variables explicatives. Prédire une probabilité de réponse nécessite donc de fixer la valeur de *toutes* les variables explicatives.

Deux méthodes peuvent être utilisées pour la prédiction (seule la seconde est disponible pour les GLMMs), les deux étant basées sur la fonction `predict()` :

— donner la valeur (ou les valeurs pour faire plusieurs prédictions) de chacune des variables explicatives directement dans la fonction, sous la forme d'une liste : `predict(modele,newdata=li-`

st(variables),type="response"), où variables est un enchaînement de variable1=valeur,variable2=valeur...

— créer un tableau contenant une colonne par variable explicative (les noms de colonnes doivent être *strictement identiques* aux noms des variables du modèle), et remplir chaque ligne en fonction des valeurs pour lesquelles la prédiction doit être faite (il y a donc une prédiction par ligne de ce tableau). Puis : predict(modele,newdata=tableau,type="response").
Remarque : pour les GLMMs, les valeurs du (ou des) facteur(s) aléatoire(s) peuvent être données ou non. Si ce n'est pas le cas, il est nécessaire d'ajouter l'argument re.form=NA à la fonction predict(). Les prédictions prennent alors en compte l'effet moyen de toutes les modalités du (ou des) facteur(s) aléatoire(s) du modèle.

EXEMPLE(S)

Avec un modèle contenant un facteur à deux niveaux (A et B), une covariable variant de 0 à 30, et leur interaction :

```
> modele <- glm(reponse~facteur*covariable,family="binomial")
```

On peut prédire une probabilité de réponse de cette façon :

```
> predict(modele,newdata=list(facteur="A",covariable=10),type="response")
```

Ou, pour plusieurs prédictions :

```
> predict(modele,newdata=list(facteur=c("A","B"),covariable=c(10,10)),type="response")
```

Ou encore :

```
> predict(modele,newdata=list(facteur=c("A","B"),covariable=rep(10,2)),type="response")
```

Ou encore créer un tableau de ce type :

```
> tableau
  facteur covariable
1       A         10
2       B         10
```

Puis :

```
> predict(modele,newdata=tableau,type="response")
```

Graphes

Effet d'un facteur

Pour illustrer l'effet d'un facteur, on réalise généralement un diagramme en barres où sont représentées les probabilités de réponse moyennes par modalité. Deux types de moyennes peuvent être représentées :

— les moyennes brutes (*i.e.* calculées à partir des données brutes), avec leurs erreurs standards : voir fiches **35** et **37**, et l'utilisation de la fonction tapply(). Attention, dans le cas des GLMMs, ces moyennes et erreurs standards ne tiennent pas compte du (des) facteur(s) aléatoire(s).

— les moyennes ajustées en fonction des autres variables du modèle, également avec leurs erreurs standards : voir fiche **43**.

Une fois les moyennes et erreurs standards récupérées, le diagramme peut être tracé (voir fiche **34**).

Relation avec une covariable

Le modèle logistique suppose une relation sigmoïde (*i.e.* en « S ») entre la variable à expliquer et chacune des covariables. Illustrer cette relation nécessite trois étapes :

1. Tracer les points observés : plot(reponse~covariable).
2. Créer un vecteur ayant les mêmes minimum et maximum que la covariable mais découpé en très petits intervalles : x <- seq2(covariable)[4].
3. Ajouter la courbe de la relation sur le graphe, ajustée par le modèle. La détermination de tous les points de la courbe est en fait basée sur une prédiction : la valeur que prend la variable à expliquer pour chaque valeur du vecteur x. C'est le grand nombre de valeurs de ce vecteur, et le très petit intervalle qui les sépare, qui donne l'aspect lissé de la courbe (qui est en fait constituée de centaines de points reliés entre eux). Si le modèle contient d'autres variables explicatives que la covariable du graphe, il faut en fixer toutes les valeurs à l'identique (de sorte que seule

la covariable du graphe change de valeur pour toutes les prédictions). La courbe s'ajoute *via* `lines(x,predict(modele,newdata=variables.à.expliquer,type="response"))`.
Remarque : la fonction `lines()` peut être utilisée plusieurs fois de suite pour tracer plusieurs courbes sur le même graphe, par exemple pour plusieurs niveaux d'un même facteur contenu dans le modèle. Pour modifier le type de tracé et la couleur des courbes, utiliser les arguments `lty` et `col` respectivement (pour plus d'informations et (beaucoup) plus d'options graphiques, voir `?par`). Enfin, pour ajouter une légende, utiliser la fonction `legend()` (voir `?legend` pour plus d'informations).

EXEMPLE(S)

On se base toujours sur le modèle suivant :

```
> modele <- glm(reponse~facteur*covariable,family="binomial")
```

Étape 1 : tracer les points correspondant aux données observées :

```
> plot(reponse~covariable)
```

Étape 2 : créer le vecteur `x` :

```
> x <- seq2(covariable)
```

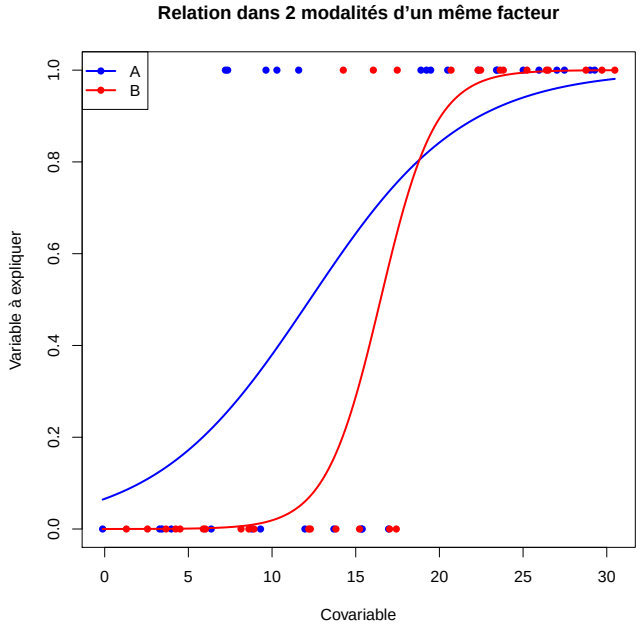
[4]

Étape 3 : ajouter la courbe. On choisit de se placer dans la modalité A du facteur :

```
> lines(x,predict(modele,newdata=list(covariable=x,facteur=rep("A",length(x))),
type="response"))
```

La fonction `rep()` est très utile dans cette situation, puisqu'elle permet de créer très simplement un vecteur de la même longueur que `x`, contenant une seule valeur répétée.
Pour ajouter la courbe de la relation dans la modalité B :

```
> lines(x,predict(modele,newdata=list(covariable=x,facteur=rep("B",length(x))),
type="response"))
```

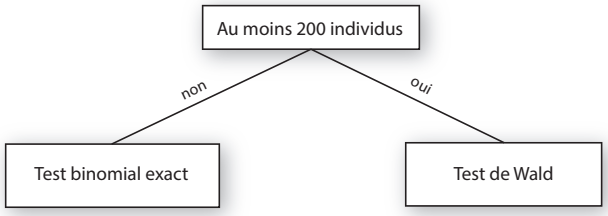


49. Conformité de plusieurs probabilités à des valeurs théoriques – plus de 2 classes

— PACKAGE(S) ADDITIONNEL(S) UTILISÉ(S) DANS CETTE FICHE —

`RVAideMemoire`[1]

Pour choisir le test approprié :



Le test binomial exact est toujours le plus fiable mais son temps de calcul augmente avec le nombre d'individus. Lorsque l'effectif est suffisamment grand, l'approximation faite par le test de Wald est assez satisfaisante pour qu'il puissent être utilisé.

Test binomial exact (non paramétrique)

Pour réaliser le test : `multinomial.theo.multcomp(reponse,p=proba.theo,prop=TRUE)`[1] où `reponse` est un facteur à au moins trois niveaux donnant la réponse de chaque individu, et `proba.theo` un vecteur donnant la probabilité théorique de chaque niveau (dans l'ordre des niveaux de `reponse`). La somme de ces probabilités doit valoir 1. Il y a en fait un test binomial exact par proportion testée.

Test de Wald (paramétrique)

Pour réaliser le test : `wald.ptheo.multinom.test(reponse,p=proba.theo)`[1]. Il y a en fait un test de Wald par proportion testée.

50. Comparaison de plusieurs probabilités – plus de 2 classes

— PACKAGE(S) ADDITIONNEL(S) UTILISÉ(S) DANS CETTE FICHE —

[1]RVAideMemoire

Test de Wald (paramétrique)

Pour réaliser le test : `prop.multinom.test(reponse)`[1] où `reponse` est un facteur à au moins trois niveaux donnant la réponse de chaque individu.

Il n'y a pas de test global dans cette situation. La fonction `prop.multinom.test()`[1] réalise en fait directement toutes les comparaisons multiples entre les différentes classes (*i.e.* les probabilités respectives de ces classes sont comparées). Il y a donc un test de Wald par comparaison.

— EXEMPLE(S) —

On a réalisé une expérimentation où chaque individu avait le choix entre trois options : Opt1, Opt2, Opt3. Les résultats sont les suivants :

```
> resultat <- factor(c("Opt1","Opt2","Opt1","Opt2","Opt2","Opt1","Opt3","Opt2",
  "Opt1","Opt2","Opt2","Opt3","Opt2","Opt2","Opt2","Opt1","Opt3","Opt2","Opt1",
  "Opt2","Opt1","Opt2","Opt1","Opt2","Opt2"))
```

La probabilité de chaque classe (et son erreur standard) est donnée par :

```
> prop.multinom(resultat)[1]
$probs
Opt1 Opt2 Opt3
0.32 0.56 0.12

$se
  Opt1   Opt2   Opt3
0.0952 0.1013 0.0663
```

On compare ces probabilités :

```
> prop.multinom.test(resultat)[1]
```

51. Analyser une réponse qualitative nominale à plus de 2 classes

— PACKAGE(S) ADDITIONNEL(S) UTILISÉ(S) DANS CETTE FICHE —

[1]`nnet`, [2]`RVAideMemoire`, [3]`car`

⚠ L'analyse présentée dans cette fiche est basée sur un modèle. Il est indispensable de maîtriser les notions des fiches **39** à **42**. ⚠

Réponse

Avec une réponse nominale à plus de 2 classes (*i.e.* définissant plus de deux catégories non ordonnées), ce qui est analysé est la *probabilité qu'un individu soit de chacune des classes.* Il sagit donc d'une réponse multiple, et non unique comme dans la plupart des situations classiques (comme l'analyse d'une réponse binaire, voir fiche **48**). Cela complique sérieusement l'interprétation des résultats.

Pour pouvoir analyser une telle réponse, il est nécessaire de définir une « classe de référence ». Car ce sur ce quoi va travailler le modèle est en fait la probabilité qu'un individu soit d'une classe donnée *plutôt que de celle de référence.* Pour K classes, on doit donc interpréter $K-1$ rapports par variable explicative, qui s'étendent à $\frac{K.(K-1)}{2}$ rapports (par variable explicative!) pour toutes les combinaisons deux-à-deux des K classes. On voit bien la complexité de travailler sur de telles réponses. D'où l'idée de se restreindre à un design expérimental très simple (*i.e.* peu de variables explicatives et encore moins d'interactions) et à un nombre de classes (*i.e.* K) limité. Pour des raisons analogues, mieux vaut éviter les variables explicatives qualitatives à plus de deux niveaux (ou alors il faut être prêt à s'accrocher pour l'interprétation, qui peut être fastidieuse délicate).

Dans **R**, la variable à expliquer doit être un facteur à plus de deux niveaux. Par défaut la 1ère modalité est définie comme classe de référence. Le choix de cette classe n'a aucune incidence sur l'analyse.

Modèle utilisé

Le modèle utilisé est un modèle multinomial (ou « polytomique non ordonné »), une extension du Modèle Linéaire Généralisé (*Generalized Linear Model* ou GLM) pour réponse binaire (voir fiche **48**).

Remarque : il existe en fait plusieurs types de modèles multinomiaux. Le cas décrit ici – le plus fréquent – est celui d'un modèle *logistique.* Lorsque toutes les variables explicatives sont des covariables, on est dans le cas particulier d'une *régression multinomiale logistique.*

Construction du modèle

Pour créer le modèle : `modele<-multinom(formule,abstol=1e-15,reltol=1e-15,maxit=10-00)`[1]. Voir fiche **40** pour une explication détaillée de la construction d'une formule. Les arguments `abstol`, `reltol` et `maxit` sont des options assurant une plus grande précision des résultats.

Remarque : la construction d'un modèle multinomial par la fonction `multinom()`[1] est un processus itératif (comme très souvent) qui a la particularité d'être en partie visible. Pour supprimer les messages affichés à la construction du modèle, ajouter l'argument `trace=FALSE`.

Vérification de la dépendance des paramètres du modèle aux données

La variable à expliquer n'étant pas quantitative, la vérification de la validité du modèle ne se fait pas de manière habituelle ici. On cherche en fait à savoir à quel point les paramètres du modèle dépendent des données qu'on lui fournit. Si les paramètres sont très sensibles à la moindre variation dans les données, c'est que le modèle est mal défini. Cela peut indiquer (entre autres) que le modèle peut être simplifié ou que certains paramètres ne sont pas calculables.

La valeur qui représente cette dépendance est le *conditionnement* de la matrice hessienne. Elle est obtenue *via* `cond.multinom(modele)`[2]. Il n'y a pas vraiment de seuil absolu, mais on considère généralement que si elle est supérieure à 10^5 c'est que le modèle est mal défini.

Test(s)

L'effet des variables explicatives est testé *via* `Anova(modele)`[3] (voir fiche **42** pour une explication détaillée des hypothèses testées). Le test réalisé est un test du rapport des vraisemblances (*Likelihood Ratio Test* ou LR Test) – en fait un test par terme du modèle (*i.e.* un par ligne du tableau renvoyé).

Si une variable explicative a un effet significatif, cela indique qu'elle influence *le rapport entre au moins deux classes* de la variable réponse. Autrement dit, la probabilité qu'un individu soit d'une classe donnée *par rapport à une autre classe donnée*. Pour identifier quel(s) rapport(s) est (sont) significativement influencé(s) par la variable explicative en question : `test.multinom(modele,variable)`[2] où `variable` est la variable explicative dont on souhaite étudier l'influence en détail. Ce qui est renvoyé par `test.multinom()`[2] dépend de la nature de la variable explicative :

— Pour une variable explicative quantitative : la fonction renvoie un tableau où chaque ligne correspond à un rapport entre deux classes de la variable réponse (la syntaxe `A|B` signifie « probabilité de A par rapport à la probabilité de B »). Le *signe* du coefficient indique le sens de la relation entre un rapport et la variable explicative. L'*odds ratio* est une façon simple de comprendre ce rapport, qui indique de combien il varie *pour une augmentation d'une unité de la variable explicative*.
Remarque : il n'est pas surprenant que les *p-values* des tests par rapport soient moins nettes que la *p-value* globale obtenue *via* `Anova(modele)`[3]. D'abord parce qu'un test global ne se réduit pas à une somme de tests individuels, mais aussi car ce ne sont pas les mêmes tests qui sont réalisés : test du rapport des vraisemblances au niveau global, tests de Wald pour chaque rapport. Le second est moins puissant que le premier (voir fiche **17**).

EXEMPLE(S)

Sur un modèle dont la réponse a trois classes (A/B/C), une covariable a un effet significatif. Le détail de son influence est le suivant :

```
      Coeff      SE Odds.ratio       z Pr(>|z|)
A|C -0.93509 0.57383     0.3926 -1.6296  0.10319
B|C  1.56595 1.05528     4.7872  1.4839  0.13783
B|A  2.50104 1.20247    12.1952  2.0799  0.03753 *
--
Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

La conclusion est que la covariable n'influence que le rapport entre les modalités A et B (seule *p-value* significative). D'après l'odds ratio, la probabilité qu'un individu soit B plutôt que A est 12.2 fois plus importante quand la covariable augmente d'une unité.
Remarque : un coefficient significativement différent de 0 revient à un odds ratio significativement différent de 1.

— Pour une variable explicative qualitative à deux niveaux : la fonction renvoie un tableau structuré comme pour une covariable. L'odds ratio d'un rapport entre deux classes de la variable réponse indique le rapport entre (i) ce rapport dans une classe de la variable explicative et (ii) ce même rapport dans l'autre classe (c'est donc un rapport de rapports). La direction du rapport entre les classes de la variable explicative est indiquée en titre du tableau.

EXEMPLE(S)

Sur un modèle dont la réponse a trois classes (A/B/C), un facteur à deux niveaux (Femelle/Mâle) a un effet significatif. Le détail de son influence est le suivant :

```
$'Male|Femelle'
      Coeff     SE Odds.ratio       z Pr(>|z|)
A|C  0.1702 1.8533    1.18550  0.09182  0.92684
B|C -1.9660 0.8645    0.14002 -2.27415  0.02296 *
B|A -5.1361 4.0226    0.00588 -1.27682  0.20166
--
Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

La conclusion est que le facteur n'influence que le rapport entre les modalités B et C. D'après l'odds ratio, la probabilité que les mâles soient B plutôt que C est seulement 0.14 fois celle des femelles (autrement dit les femelles ont $\frac{1}{0.14} = 7.14$ fois plus de chances d'être B plutôt que C par rapport aux mâles).

— Pour une variable explicative qualitative à plus de deux niveaux : la fonction renvoie un tableau *par rapport possible entre les classes de la variable explicative*. C'est donc une extension du cas précédent, forcément bien plus difficile à interpréter.

Prédiction à partir du modèle

L'intérêt des modèles est à la fois de comprendre l'importance des différentes variables explicatives sur la variable à expliquer, mais également de prédire la valeur que prendrait cette variable à expliquer pour des valeurs *connues* des variables explicatives. Prédire une série de probabilités nécessite donc de fixer la valeur de *toutes* les variables explicatives.

Deux méthodes peuvent être utilisées pour la prédiction, les deux étant basées sur la fonction `predict()` :

— donner la valeur (ou les valeurs pour faire plusieurs prédictions) de chacune des variables explicatives directement dans la fonction, sous la forme d'une liste : `predict(modele,newdata=list(variables))`, où `variables` est un enchaînement de `variable1=valeur,variable2=valeur...`

— créer un tableau contenant une colonne par variable explicative (les noms de colonnes doivent être *strictement identiques* aux noms des variables du modèle), et remplir chaque ligne en fonction des valeurs pour lesquelles la prédiction doit être faite (il y a donc une prédiction par ligne de ce tableau). Puis : `predict(modele,newdata=tableau)`.

La fonction `predict()` renvoie, pour chaque prédiction, la classe la plus probable de la variable réponse sachant la valeur des variables explicatives qui ont servi à la prédiction. Il est possible d'obtenir le détail des probabilités pour chacune des classes de la variable réponse en ajoutant l'arguments `type="probs"`. Le résultat est alors une matrice où chaque ligne est une prédiction et chaque colonne une classe de la variable réponse.

EXEMPLE(S)

Avec un modèle contenant un facteur à deux niveaux (A et B) et une covariable variant de 0 à 30 :

```
> modele <- multinom(reponse~facteur+covariable,abstol=1e-15,reltol=1e-15,maxit=1000)[1]
```

On peut prédire la classe la plus probable de cette façon :

```
> predict(modele,newdata=list(facteur="A",covariable=10))
```

Ou, pour plusieurs prédictions :

```
> predict(modele,newdata=list(facteur=c("A","B"),covariable=c(10,10)))
```

Ou encore :

```
> predict(modele,newdata=list(facteur=c("A","B"),covariable=rep(10,2)))
```

Ou encore créer un tableau de ce type :

```
> tableau
  facteur covariable
1       A         10
2       B         10
```

Puis :

```
> predict(modele,newdata=tableau)
```

52. Analyser un rang dans un classement

— PACKAGE(S) ADDITIONNEL(S) UTILISÉ(S) DANS CETTE FICHE —

[1]`ordinal`, [2]`car`, [3]`RVAideMemoire`, [4]`emmeans`

⚠ L'analyse présentée dans cette fiche est basée sur un modèle. Il est indispensable de maîtriser les notions des fiches **39** à **42**. ⚠

Réponse

Un rang dans un classement (appelé « note »dans cette fiche) est une variable piégeuse, surtout s'il est codé numériquement (*i.e.* 1/2/3/...). On pourrait en effet penser qu'il s'agit d'une variable quantitative, or il n'en est rien puisque c'est une variable qualitative ordinale (voir fiche **11**). La preuve : les notes (1/2/3) peuvent parfaitement être remplacées par des lettres (A/B/C) ou des expressions (bon/moyen/mauvais) sans changer le sens de la variable. On parle parfois de variables « semi-quantitatives » car elles supposent un processus sous-jacent quantitatif, mais elles ne le sont pas elles-mêmes. C'est pour cette raison que le paramètre de position (voir fiche **35**) le plus pertinent qui leur correspond est la *médiane*, et non la moyenne.

Les notes doivent être placés dans un type d'objet particulier, un *facteur ordonné*. Pour créer cet objet : `notes<-factor(serie,levels=classement,ordered=TRUE)` où `serie` est la série de données et `classement` un vecteur donnant le classement des notes dans l'*ordre croissant* (entre guillemets).

— EXEMPLE(S) —

Si les notes sont codées numériquement 1/2/3/4 (1 étant le rang le plus haut) :

```
> notes <- factor(serie,levels=c("4","3","2","1"),ordered=TRUE)
```

Lorsque l'on affiche le vecteur `notes`, les niveaux sont donnés sous cette forme :

```
Levels: 4 < 3 < 2 < 1
```

Si les notes sont codées en expression bon/moyen/mauvais (bon étant le rang le plus haut) :

```
> notes <- factor(serie,levels=c("mauvais","moyen","bon"),ordered=TRUE)
```

Lorsque l'on affiche le vecteur `notes`, les niveaux sont donnés sous cette forme :

```
Levels: mauvais < moyen < bon
```

Modèle utilisé

Si le principe de l'analyse est le même qu'il y ait des séries appariées ou non, quelques modalités diffèrent. La première est que sans séries appariées le modèle utilisé est un Modèle à Odds Proportionnels (*Proportional Odds Model* ou POM), tandis qu'avec des séries appariées c'est un Modèle à Odds Proportionnels Mixte (*Proportional Odds Mixed Model* ou POMM ; « mixte » sous-entendant « contenant au moins un facteur aléatoire », facteurs que l'on utilise précisément pour identifier les séries appariées (voir fiche **11**)).

Ces modèles sont une sorte d'extension de ceux permettant d'analyser une réponse binaire (voir fiche **48**), mais ne rentrent pas dans la famille des Modèles Linéaires Généralisés (*Generalized Linear Models* ou GLMs). Dans le cas de notes, l'analyse consiste en fait à étudier la *probabilité de passer d'une note à la note supérieure, et ce pour chaque transition du classement.*

Remarque : les POM(M)s sont en fait un cas particulier des Modèles à Lien Cumulatif (*Cumulative Link (Mixed) Models* ou CLM(M)s). Le cas décrit ici – le plus fréquent – est celui d'un modèle *logistique.*

Construction du modèle

Pour créer le modèle :

— sans séries appariées : `modele<-clm(formule)`[1]
— avec des séries appariées : `modele<-clmm(formule)`[1].

Voir fiche **40** pour une explication détaillée de la construction d'une formule. De manière générale, on peut dire que :

— inclure un facteur permet de tester si les notes diffèrent entre les niveaux de ce facteur.
— inclure une covariable permet de tester s'il existe une relation entre cette covariable et la variable à expliquer.
— inclure une interaction entre deux variables explicatives permet de tester si l'effet de l'une dépend de l'autre (le raisonnement est le même pour une interaction incluant plus de deux termes). Le cas particulier d'une interaction entre une covariable et un facteur permet de tester si la relation entre la covariable et la variable à expliquer est différente selon la modalité du facteur.

Vérification de la dépendance des paramètres du modèle aux données

La variable à expliquer n'étant pas quantitative, la vérification de la validité du modèle ne se fait pas de manière habituelle ici. On cherche en fait à savoir à quel point les paramètres du modèle dépendent des données qu'on lui fournit. Si les paramètres sont très sensibles à la moindre variation dans les données, c'est que le modèle est mal défini. Cela peut indiquer (entre autres) que le modèle peut être simplifié ou que certains paramètres ne sont pas calculables.

La valeur qui représente cette dépendance est le *conditionnement* de la matrice hessienne. Elle est obtenue simplement en appelant `modele`, sous l'intitulé `cond.H`. Il n'y a pas vraiment de seuil absolu, mais on considère généralement que si elle est supérieure à 10^5 c'est que le modèle est mal défini.

Test(s)

Quel que soit le modèle, l'effet des variables explicatives est testé par la même fonction : `Anova(modele)`[2] (voir fiche **42** pour une explication détaillée des hypothèses testées). Le test réalisé est un test du rapport des vraisemblances (*Likelihood Ratio Test* ou LR Test) – en fait un test par terme du modèle (*i.e.* un par ligne du tableau renvoyé).

Remarque : pour pouvoir utiliser la fonction `Anova()`[2] sur un modèle créé par `clm()`[1] ou `clmm()`[1] il est nécessaire d'avoir chargé le package `RVAideMemoire`.

Si un facteur (ou une interaction impliquant un facteur) a un effet significatif, il est nécessaire de réaliser des comparaisons multiples pour identifier les modalités (ou combinaisons de modalités) qui diffèrent. Voir fiche **43** pour réaliser ces comparaisons.

Si une covariable a un effet significatif, la direction de son effet est donnée par le *signe* du paramètre associé. Les valeurs de tous les paramètres du modèle sont obtenues *via* `summary(modele)`. Elles sont appelées `Estimate` et se trouvent dans le tableau `Coefficients`. Si le coefficient portant le nom de la covariable est négatif, la note diminue quand la valeur de la covariable augmente ; s'il est positif, la note augmente quand la valeur de la covariable augmente.

Prédiction à partir du modèle (non disponible pour les POMMs)

L'intérêt des modèles est à la fois de comprendre l'importance des différentes variables explicatives sur la variable à expliquer, mais également de prédire la valeur que prendrait cette variable à expliquer pour des valeurs *connues* des variables explicatives. Prédire une note nécessite donc de fixer la valeur de *toutes* les variables explicatives.

Deux méthodes peuvent être utilisées pour la prédiction, les deux étant basées sur la fonction `predict()` :

— donner la valeur (ou les valeurs pour faire plusieurs prédictions) de chacune des variables explicatives directement dans la fonction, sous la forme d'une liste : `predict(modele,newdata=list(variables))`, où `variables` est un enchaînement de `variable1=valeur,variable2=valeur...`
— créer un tableau contenant une colonne par variable explicative (les noms de colonnes doivent être *strictement identiques* aux noms des variables du modèle), et remplir chaque ligne en fonction des valeurs pour lesquelles la prédiction doit être faite (il y a donc une prédiction par ligne de ce tableau). Puis : `predict(modele,newdata=tableau)`.

La spécificité des POM(M)s (et plus globalement des CLM(M)s) est qu'ils travaillent sur plusieurs valeurs simultanément, *i.e.* les probabilités associées *à chaque note* du classement. La fonction `predict()` renvoie donc, pour chaque prédiction, autant de valeurs qu'il y a de notes possibles. Ces valeurs correspondent à la probabilité respective de chaque note, sachant la valeur des variables explicatives qui ont

servi à la prédiction. Les prédictions sont organisées en une matrice (qui est en fait dans le compartiment `fit` de la liste renvoyée par `predict()`) où chaque ligne est une prédiction et chaque colonne une note du classement (du bas vers le haut du classement).

Deux autres types de prédiction sont disponibles :

— les probabilités cumulées, en ajoutant l'argument `type="cum.prob"` à la fonction `predict()`. La fonction renvoie également une matrice (dans le compartiment `cprob1` d'une liste) avec une ligne par prédiction et une colonne par note. Chaque colonne contient non pas la probabilité de la note correspondante, mais la probabilité que la note obtenue soit *inférieure ou égale* à cette note. La dernière colonne (*i.e.* la note la plus élevée du classement) contient donc nécessairement la valeur 1.

— la note la plus probable, en ajoutant l'argument `type="class"` à la fonction `predict()`. La fonction renvoie cette fois un vecteur (dans le compartiment `fit` d'une liste) contenant une valeur par prédiction. Cette valeur est la note la plus probable sachant la valeur des variables explicatives qui ont servi à la prédiction.

EXEMPLE(S)

Avec un modèle contenant un facteur à deux niveaux (A et B), une covariable variant de 0 à 30, et leur interaction :

```
> modele <- clm(reponse~facteur*covariable)[1]
```

On peut prédire une note de cette façon :

```
> predict(modele,newdata=list(facteur="A",covariable=10))
```

Ou, pour plusieurs prédictions :

```
> predict(modele,newdata=list(facteur=c("A","B"),covariable=c(10,10)))
```

Ou encore :

```
> predict(modele,newdata=list(facteur=c("A","B"),covariable=rep(10,2)))
```

Ou encore créer un tableau de ce type :

```
> tableau
  facteur covariable
1       A         10
2       B         10
```

Puis :

```
> predict(modele,newdata=tableau)
```

Graphes

Effet d'un facteur

Pour illustrer l'effet d'un facteur, on réalise généralement un diagramme en barres (voir fiche **34**). Mais du fait de la complexité d'une variable à expliquer telle que des notes, plusieurs représentations sont possibles. On a également le choix de l'information que l'on veut représenter : la fréquence observée des différentes notes (dans le jeu de données) ou la probabilité ajustée de ces notes (calculée à partir du modèle). Il est souvent plus pertinent de représenter les probabilités ajustées, car elles illustrent la variation réellement due au facteur, après avoir retiré la variation dues aux autres variables explicatives du modèles. De plus, seules les probabilités ajustées prennent en compte l'effet des facteurs aléatoires (s'il y en a dans l'analyse).

Les représentations possibles sont :

— la fréquence / probabilité de chaque note.
Les fréquences observées sont obtenues *via* `rating.prob(notes,facteur)`[3] (`facteur` peut être une interaction entre deux facteurs, spécifiée par `facteur1:facteur2`).
Deux étapes sont nécessaires pour récupérer les probabilités ajustées. D'abord calculer ces probabilités et les stocker dans un objet (appelé `EMM` ici) : `EMM <- emmeans(modele,~facteur|cut, mode="linear.predictor")`[4] (voir fiche **43** pour une explication de la syntaxe de la fonction `emmeans()`[4]; il est indispensable que la formule se termine par `|cut`). Ensuite récupérer les probabilités, qui ne sont pas données directement par la fonction `emmeans()`[4] : `rating.emmeans(EMM)`[3].

— la fréquence cumulée / probabilité cumulée de chaque note (*i.e.* la probabilité que la note soit *inférieure ou égale* à chaque note possible, du bas vers le haut du classement).
Les fréquences cumulées observées sont obtenues *via* `rating.prob(notes,facteur,type="cumprob")`[3].

Les probabilités ajustées sont récupérées (à partir du même objet `EMM`) *via* `rating.emmeans(EMM, type="cumprob")`[3].

— la note la plus fréquente / probable. Le diagramme en barres ne peut pas être utilisé dans ce cas car la valeur est qualitative et non quantitative.
La note la plus fréquente est obtenue *via* `rating.prob(notes,facteur,type="class")`[3].
La note la plus probable est obtenue (à partir du même objet `EMM`) *via* `rating.emmeans(EMM,facteur,type="class1")`[3].

53. Conformité d'une série d'effectifs à une distribution théorique

— PACKAGE(S) ADDITIONNEL(S) UTILISÉ(S) DANS CETTE FICHE —

[1]`car`, [2]`dgof`, [3]`RVAideMemoire`

L'ajustement à une distribution théorique se teste avant tout *graphiquement*, car il est rare que des données réelles suivent parfaitement une loi théorique. Il y a donc toujours une part de subjectivité dans l'estimation d'un ajustement, que les tests statistiques ne peuvent pas refléter.

Test graphique

Le test visuel est réalisé grâce au graphe quantile-quantile, tracé de la manière suivante : `qqPlot(serie,dist="loi",par)`[1] où `serie` est un vecteur contenant la série de données, `loi` est la loi théorique choisie (entre guillemets) et `par` ses paramètres séparés par une virgule (voir fiches **19** à **22**).

— EXEMPLE(S) —

Ajustement à une loi binomiale (voir fiche **20**) :

```
> qqPlot(serie,dist="binom",n,p)[1]
```

Ajustement à une loi de Poisson (voir fiche **21**) :

```
> qqPlot(serie,dist="pois",lambda)[1]
```

La distribution de la série suit la loi théorique choisie si les points du graphe sont à peu près alignés sur une droite. Toute autre structuration des points (courbure(s), nombreux points éloignés...) indique le contraire. La droite tracée est celle qui passe par les 1er et 3ème quartiles de la distribution représentée, et son intervalle de confiance est affiché en pointillés. Les points peuvent ne pas être parfaitement alignés sur la droite, mais s'ils restent dans l'intervalle de confiance l'ajustement est considéré comme correct.

Test statistique

Test de Cramér - von Mises (non paramétrique)

Pour réaliser le test : `cvm.test(serie,cdf.discrete(serie,"loi",par)`[3]`)`[2].

— EXEMPLE(S) —

Ajustement à une loi binomiale (voir fiche **20**) :

```
> cvm.test(serie,cdf.discrete(serie,"binom",n,p)[3])[2]
```

Ajustement à une loi de Poisson (voir fiche **21**) :

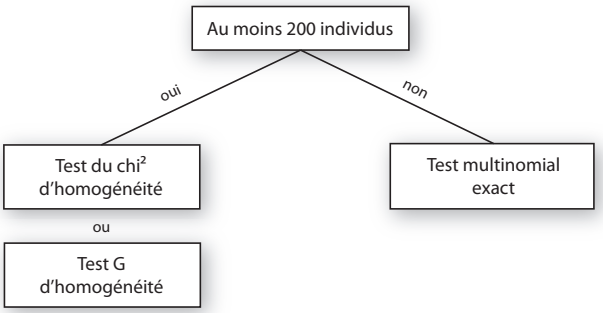
```
> cvm.test(serie,cdf.discrete(serie,"pois",lambda)[3])[2]
```

54. Comparaison de plusieurs effectifs

— PACKAGE(S) ADDITIONNEL(S) UTILISÉ(S) DANS CETTE FICHE —

[1]`RVAideMemoire`

Pour choisir le test approprié :



Le test exact est toujours le plus fiable mais son temps de calcul augmente grandement avec le nombre d'individus. Lorsque l'effectif total est suffisamment grand, l'approximation faite par les tests du χ^2 et G est assez satisfaisante pour qu'ils puissent être utilisés. Les résultats de ces deux tests sont très semblables ; choisir entre l'un et l'autre relève plus d'une habitude que d'une raison statistique.

Test multinomial exact (non paramétrique)

Pour réaliser le test : `multinomial.test(effectifs)`[1] où `effectifs` est un vecteur contenant les effectifs.

Une *p-value* significative indique qu'au moins deux effectifs diffèrent l'un de l'autre, sans préciser lesquels. Il est dans ce cas nécessaire de réaliser des comparaisons deux-à-deux pour identifier les effectifs en question, *via* `multinomial.multcomp(effectifs)`[1].

Il peut arriver que les comparaisons deux-à-deux n'indiquent aucune différence significative, contrairement au test global. Dans ce cas, la solution la plus prudente est de considérer qu'on ne peut pas savoir quels effectifs sont responsables du rejet de l'hypothèse nulle dans le test global.

Test du χ^2 d'homogénéité (non paramétrique)

Pour réaliser le test : `chisq.test(effectifs)`.

Si la *p-value* est significative, les comparaisons deux-à-deux sont réalisées *via* `chisq.multcomp(effectifs)`[1].

Test G d'homogénéité (non paramétrique)

Pour réaliser le test : `G.test(effectifs)`[1].

Si la *p-value* est significative, les comparaisons deux-à-deux sont réalisées *via* `G.multcomp(effectifs,p=prop.theo)`[1].

55. Analyser un décompte d'individus

— PACKAGE(S) ADDITIONNEL(S) UTILISÉ(S) DANS CETTE FICHE —

[1]`lme4`, [2]`RVAideMemoire`, [3]`MASS`, [4]`MuMIn`, [5]`car`

⚠ L'analyse présentée dans cette fiche est basée sur un modèle. Il est indispensable de maîtriser les notions des fiches **39** à **42**. ⚠

Réponse

La réponse (appelée « effectif » dans cette fiche) ne peut être que nulle ou entière positive.

Modèle utilisé

Si le principe de l'analyse est le même qu'il y ait des séries appariées ou non, quelques modalités diffèrent. La première est que sans séries appariées le modèle utilisé est un Modèle Linéaire Généralisé (*Generalized Linear Model* ou GLM), tandis qu'avec des séries appariées c'est un Modèle Linéaire Généralisé Mixte (*Generalized Linear Mixed Model* ou GLMM ; « mixte » sous-entendant « contenant au moins un facteur aléatoire », facteurs que l'on utilise précisément pour identifier les séries appariées (voir fiche **11**)).

Contrairement au Modèle Linéaire (et sa variante Mixte, voire fiche **76**), les GLM(M)s ne sont pas forcément basés sur une loi normale. Dans le cas d'effectifs, la loi à utiliser *a priori* est une loi de Poisson.

Remarque : il existe en fait plusieurs types de GLM(M)s basés sur une loi de Poisson. Le cas décrit ici – le plus fréquent – est celui d'un modèle *log-linéaire*. Lorsque toutes les variables explicatives sont des covariables, on est dans le cas particulier d'une *régression log-linéaire* ou *régression de Poisson*.

Construction du modèle

Pour créer le modèle :

— sans séries appariées : `modele<-glm(formule,family="poisson")`
— avec des séries appariées : `modele<-glmer(formule,family="poisson")`[1].

Voir fiche **40** pour une explication détaillée de la construction d'une formule. De manière générale, on peut dire que :

— inclure un facteur permet de tester si les effectifs diffèrent entre les niveaux de ce facteur.
— inclure une covariable permet de tester s'il existe une relation entre cette covariable et la variable à expliquer.
— inclure une interaction entre deux variables explicatives permet de tester si l'effet de l'une dépend de l'autre (le raisonnement est le même pour une interaction incluant plus de deux termes). Le cas particulier d'une interaction entre une covariable et un facteur permet de tester si la relation entre la covariable et la variable à expliquer est différente selon la modalité du facteur.

Vérification de l'ajustement aux données

Avant d'aller plus loin, il est indispensable de vérifier que le modèle s'ajuste bien aux données. Cette étape est fondamentale, et ce pour tout modèle, car un test basé sur un modèle mal ajusté n'est simplement pas fiable (ni valide). Deux conditions sont à vérifier (dans cet ordre).

1. Indépendance entre les résidus du modèle et les valeurs prédites

Voir fiche **41** pour une explication détaillée et la manière de vérifier cette condition. Si elle n'est pas respectée de façon flagrante, plusieurs solutions sont possibles (à tester dans cet ordre) :

1. Changer la *fonction de lien* sur laquelle est basé le modèle. Par défaut cette fonction de lien est le log et n'est pas précisée (`family="poisson"` est un raccourci pour `family=poisson(link="log")`). Tester les fonctions de lien racine carrée (`family=poisson(link="sqrt")`) ou identité (`family=poisson(link="identity")`), *i.e.* les deux autres acceptées par la loi de Poisson.
   Remarque : dans les deux cas, le modèle n'est plus un modèle log-linéaire.

2. Si certaines variables *explicatives* sont quantitatives, en transformer une ou plusieurs ($\sqrt{x}$ et $log(x)$ sont les transformations les plus fréquentes).
3. Transformer la variable *à expliquer*. Sur des effectifs, trois options sont classiques : $\sqrt{x}$, $log(x)$ si la première transformation n'est pas suffisante, $\sqrt[4]{x}$ ($= x^{1/4}$) en alternative à la deuxième s'il y a des 0 dans les données. Une fois la variable à expliquer transformée, elle ne représente plus des effectifs mais une variable continue. On change donc de modèle, et l'analyse devient celle présentée dans la fiche **76**.

Si le modèle est toujours un GLM(M) (*i.e.* si la variable à expliquer n'a pas été transformée), on passe à la condition suivante.

2. Absence de sur-dispersion des résidus

Voir fiche **41** pour une explication détaillée. Pour vérifier cette condition, il suffit de calculer le rapport entre la *déviance résiduelle* (*residual deviance*) du modèle et ses *degrés de liberté résiduels* (*residual degrees of freedom*). Si le rapport est > 1.5, il y a sur-dispersion. Pour obtenir ces valeurs :

— GLM : appeler `summary(modele)` et repérer la ligne `Residual deviance: xx.xx on xx degrees of freedom`
— GLMM : utiliser `overdisp.glmer(modele)`[2].

S'il y a sur-dispersion, plusieurs solutions sont possibles :

— GLM : deux alternatives sont envisageables :
  — Remplacer la loi de Poisson par une loi quasi-Poisson, *via* `family="quasipoisson"`. La fonction de lien par défaut est le log, si elle a été modifiée à l'étape précédente il faut garder celle qui a été sélectionnée (par exemple `family=quasipoisson(link="sqrt")`).
  — Remplacer la loi de Poisson par une loi binomiale négative. Il faut recréer le modèle avec une autre fonction : `modele<-glm.nb(formule)`[3]. La fonction de lien par défaut est le log, si elle a été modifiée à l'étape précédente il faut garder celle qui a été sélectionnée en utilisant l'argument `link` (par exemple `modele<-glm.nb(formule,link=sqrt)`).

  La loi binomiale négative donne souvent de très bons résultats. Si elle ne suffit pas, la loi quasi-Poisson peut être utilisée. Cette dernière règlera toujours le problème de la sur-dispersion des résidus.
— GLMM :
  — Si la fonction de lien n'a pas été changée à l'étape précédente : remplacer la loi de Poisson par une loi binomiale négative. Il faut recréer le modèle, avec une autre fonction : `modele<-glmer.nb(formule)`[1].
  — Si la fonction de lien a été changée à l'étape précédente : ajouter un facteur aléatoire au modèle, dont chaque niveau correspond à un individu (statistique). Cela se fait en deux étapes :
    1. Créer le facteur aléatoire : `obs<-factor(1:length(reponse))` où `reponse` est la variable à expliquer.
    2. Recréer le modèle en ajoutant à la fin de la formule `+(1|obs)`.

Capacité explicative globale

On peut estimer la capacité explicative globale d'un modèle grâce au *coefficient de détermination* (R^2), qui représente la proportion de la variance de la variable à expliquer qui est expliquée par les variables explicatives. Ce coefficient est obtenu *via* `r.squaredGLMM(modele)`[4] (ligne `trigamma`, ou à défaut ligne `delta`). La fonction renvoie en fait deux valeurs : le R^2 *marginal* (`R2m`) qui correspond à la part de la variance expliquée uniquement par les facteurs fixes et covariables, et le R^2 *conditionnel* (`R2c`) qui correspond à la part de la variance expliquée par l'ensemble des variables explicatives (fixes et aléatoires). Dans le cas d'un GLM les deux valeurs sont identiques puisqu'il n'y a pas de facteur aléatoire.

Test(s)

Pour tous les modèles évoqués à l'exception de ceux avec une loi quasi-Poisson, l'effet des variables explicatives est testé par la même fonction : `Anova(modele)`[5] (voir fiche **42** pour une explication détaillée des hypothèses testées). Cependant, ce ne sont pas les mêmes tests qui sont réalisés selon le modèle :

— GLM : la fonction réalise un test du rapport des vraisemblances (*Likelihood Ratio Test* ou LR Test) – en fait un test par terme du modèle (*i.e.* un par ligne du tableau renvoyé).

— GLMM : la fonction réalise un test de Wald – en fait un test par terme du modèle (*i.e.* un par ligne du tableau renvoyé).

Dans le cas d'un GLM avec une loi quasi-Poisson, le test à utiliser est un test F. Il est réalisé *via* `Anova(modele,test="F")`[5].

Si un facteur (ou une interaction impliquant un facteur) a un effet significatif, il est nécessaire de réaliser des comparaisons multiples pour identifier les modalités (ou combinaisons de modalités) qui diffèrent. Voir fiche **43** pour réaliser ces comparaisons.

Si une covariable a un effet significatif, la direction de son effet est donnée par le *signe* du paramètre associé. Les valeurs de tous les paramètres du modèle sont obtenues *via* `summary(modele)`. Elles sont appelées `Estimate` et se trouvent dans le tableau `Coefficients` pour un GLM, `Fixed effects` pour un GLMM. Si le coefficient portant le nom de la covariable est négatif, l'effectif diminue quand la valeur de la covariable augmente ; s'il est positif, l'effectif augmente quand la valeur de la covariable augmente.

Prédiction à partir du modèle

L'intérêt des modèles est à la fois de comprendre l'importance des différentes variables explicatives sur la variable à expliquer, mais également de prédire la valeur que prendrait cette variable à expliquer pour des valeurs *connues* des variables explicatives. Prédire un effectif nécessite donc de fixer la valeur de *toutes* les variables explicatives.

Deux méthodes peuvent être utilisées pour la prédiction (seule la seconde est disponible pour les GLMMs), les deux étant basées sur la fonction `predict()` :

— donner la valeur (ou les valeurs pour faire plusieurs prédictions) de chacune des variables explicatives directement dans la fonction, sous la forme d'une liste : `predict(modele,newdata=list(variables),type="response")`, où `variables` est un enchaînement de `variable1=valeur,variable2=valeur...`

— créer un tableau contenant une colonne par variable explicative (les noms de colonnes doivent être *strictement identiques* aux noms des variables du modèle), et remplir chaque ligne en fonction des valeurs pour lesquelles la prédiction doit être faite (il y a donc une prédiction par ligne de ce tableau). Puis : `predict(modele,newdata=tableau,type="response")`.
Remarque : pour les GLMMs, les valeurs du (ou des) facteur(s) aléatoire(s) peuvent être données ou non. Si ce n'est pas le cas, il est nécessaire d'ajouter l'argument `re.form=NA` à la fonction `predict()`. Les prédictions prennent alors en compte l'effet moyen de toutes les modalités du (ou des) facteur(s) aléatoire(s) du modèle.

EXEMPLE(S)

Avec un modèle contenant un facteur à deux niveaux (A et B), une covariable variant de 0 à 30, et leur interaction :

```
> modele <- glm(reponse~facteur*covariable,family="poisson")
```

On peut prédire un effectif de cette façon :

```
> predict(modele,newdata=list(facteur="A",covariable=10),type="response")
```

Ou, pour plusieurs prédictions :

```
> predict(modele,newdata=list(facteur=c("A","B"),covariable=c(10,10)),type="response")
```

Ou encore :

```
> predict(modele,newdata=list(facteur=c("A","B"),covariable=rep(10,2)),type="response")
```

Ou encore créer un tableau de ce type :

```
> tableau
  facteur covariable
1       A         10
2       B         10
```

Puis :

```
> predict(modele,newdata=tableau,type="response")
```

Graphes

Effet d'un facteur

Pour illustrer l'effet d'un facteur, on réalise généralement un diagramme en barres où sont représentés les effectifs moyens par modalité. Deux types de moyennes peuvent être représentées :

— les moyennes brutes (*i.e.* calculées à partir des données brutes), avec leurs erreurs standards : voir fiches **35** et **37**, et l'utilisation de la fonction `tapply()`. Attention, dans le cas des GLMMs, ces moyennes et erreurs standards ne tiennent pas compte du (des) facteur(s) aléatoire(s).

— les moyennes ajustées en fonction des autres variables du modèle, également avec leurs erreurs standards : voir fiche **43**.

Une fois les moyennes et erreurs standards récupérées, le diagramme peut être tracé (voir fiche **34**).

Relation avec une covariable

Illustrer la relation entre la variable à expliquer et une covariable nécessite trois étapes :

1. Tracer les points observés : `plot(reponse~covariable)`.
2. Créer un vecteur ayant les mêmes minimum et maximum que la covariable mais découpé en très petits intervalles : `x <- seq2(covariable)`[2].
3. Ajouter la courbe de la relation sur le graphe, ajustée par le modèle. La détermination de tous les points de la courbe est en fait basée sur une prédiction : la valeur que prend la variable à expliquer pour chaque valeur du vecteur `x`. C'est le grand nombre de valeurs de ce vecteur, et le très petit intervalle qui les sépare, qui donne l'aspect lissé de la courbe (qui est en fait constituée de centaines de points reliés entre eux). Si le modèle contient d'autres variables explicatives que la covariable du graphe, il faut en fixer toutes les valeurs à l'identique (de sorte que seule la covariable du graphe change de valeur pour toutes les prédictions). La courbe s'ajoute *via* `lines(x,predict(modele,newdata=variables.à.expliquer,type="response"))`.
   Remarque : la fonction `lines()` peut être utilisée plusieurs fois de suite pour tracer plusieurs courbes sur le même graphe, par exemple pour plusieurs niveaux d'un même facteur contenu dans le modèle. Pour modifier le type de tracé et la couleur des courbes, utiliser les arguments `lty` et `col` respectivement (pour plus d'informations et (beaucoup) plus d'options graphiques, voir `?par`). Enfin, pour ajouter une légende, utiliser la fonction `legend()` (voir `?legend` pour plus d'informations).

EXEMPLE(S)

On se base toujours sur le modèle suivant :

```
> modele <- glm(reponse~facteur*covariable,family="poisson")
```

Étape 1 : tracer les points correspondant aux données observées :

```
> plot(reponse~covariable)
```

Étape 2 : créer le vecteur `x` :

```
> x <- seq2(covariable)^2
```

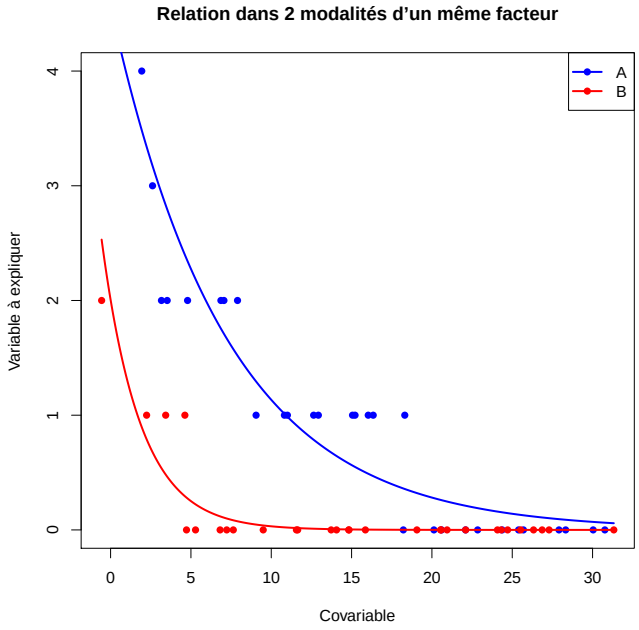
Étape 3 : ajouter la courbe. On choisit de se placer dans la modalité A du facteur :

```
> lines(x,predict(modele,newdata=list(covariable=x,facteur=rep("A",length(x))),
type="response"))
```

La fonction `rep()` est très utile dans cette situation, puisqu'elle permet de créer très simplement un vecteur de la même longueur que `x`, contenant une seule valeur répétée.

Pour ajouter la courbe de la relation dans la modalité B :

```
> lines(x,predict(modele,newdata=list(covariable=x,facteur=rep("B",length(x))),
type="response"))
```

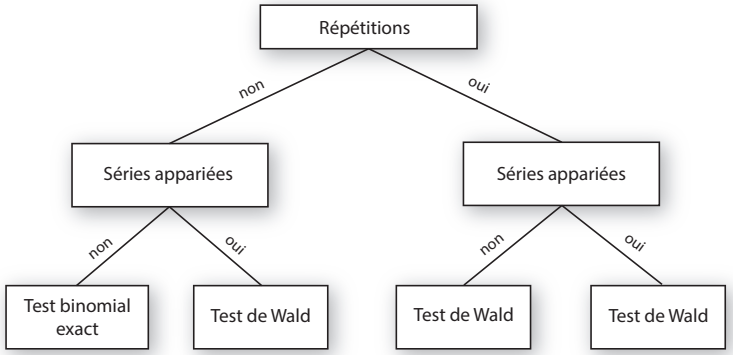

Relation dans 2 modalités d'un même facteur
Variable à expliquer
Covariable
A
B
0
1
2
3
4
0
5
10
15
20
25
30

56. Conformité d'une proportion à une valeur théorique

PACKAGE(S) ADDITIONNEL(S) UTILISÉ(S) DANS CETTE FICHE

[1]`RVAideMemoire`

Pour choisir le test approprié :



Pas de répétitions

Test binomial exact (non paramétrique)

Pour réaliser le test : `binom.test(n1,n,p=prop.theo)` où `n1` est le nombre d'individus de la catégorie d'intérêt, `n` l'effectif total et `prop.theo` la proportion théorique de la catégorie d'intérêt (0.5 par défaut).

EXEMPLE(S)

On veut comparer le sex-ratio (ici la proportion de femelles) d'un échantillon de 20 individus contenant 7 femelles et 13 mâles à un sex-ratio équilibré (donc une proportion de 0.5) :

```
> binom.test(7,20,p=0.5)
```

ou plus simplement puisque `p=0.5` par défaut :

```
> binom.test(7,20)
```

Test de Wald (paramétrique)

Pour réaliser le test : `wald.ptheo.test(reponse,blocs,p=prop.theo)`[1] où `reponse` est la réponse de chaque individu (*i.e.* à quelle catégorie il appartient, sous forme numérique ou d'un facteur) et `blocs` un facteur (aléatoire) contenant le groupe de chaque individu (dans le même ordre que `reponse`). Si `reponse` est codée sous forme 0/1, la proportion du groupe 1 est testée ; si `reponse` est un facteur, la proportion de la 2nde modalité est testée.

Répétitions

La réponse doit être une matrice à deux colonnes où chaque ligne est une répétition. Pour chacune de ces répétitions on a un décompte d'individus dans les deux catégories (une colonne par catégorie). La proportion testée est celle définie par la colonne de gauche.

Test de Wald sans séries appariées (paramétrique)

Pour réaliser le test : `wald.ptheo.test(reponse,p=prop.theo)`[1] où `reponse` est la matrice à deux colonnes.

Test de Wald avec séries appariées (paramétrique)

Pour réaliser le test : `wald.ptheo.test(reponse,blocs,p=prop.theo)`[1] où `blocs` un facteur (aléatoire) contenant le groupe de chaque répétition (*i.e.* il y a autant de valeurs dans le vecteur `blocs` que de lignes dans la matrice `reponse`).

57. Conformité de plusieurs proportions à des valeurs théoriques – 2 catégories

— PACKAGE(S) ADDITIONNEL(S) UTILISÉ(S) DANS CETTE FICHE —

`RVAideMemoire`[1]

Les données doivent être présentées sous forme d'une matrice à deux colonnes (appelée « tableau de contingence »), où les lignes correspondent aux échantillons/populations et les colonnes aux deux catégories :

	Catégorie 1	**Catégorie 2**
Éch/pop 1		
…		
Éch/pop *k*		

où sur chaque ligne on a une décompte d'individus dans les deux catégories.
Les proportions testées sont celles de la colonne de gauche (*i.e.* la catégorie 1).

Test du χ^2 de conformité (non paramétrique)

Conditions : les effectifs théoriques doivent tous être non nuls et 80 % d'entre eux doivent être ≥ 5 (« règle de Cochran », voir ci-dessous).

Les effectifs théoriques sont obtenus *via* `chisq.exp(tab.cont,p=prop.theo)`[1] où `tab.cont` est le tableau de contingence et `prop.theo` un vecteur donnant une proportion théorique par échantillon/population (de 1 à k). Les effectifs théoriques permettent de vérifier si la règle de Cochran est respectée.

Remarque : les proportions théoriques sont *indépendantes* entre les échantillons/populations. Elles n'ont pas à donner une somme de 1, puisque les échantillons/populations ne sont pas comparé(e)s entre eux (elles), mais chaque échantillon/population est testé(e) pour sa propre proportion théorique.

Pour réaliser le test : `prop.test(tab.cont,p=prop.theo)`.

Remarque : la fonction `prop.test()` n'utilise pas la règle de Cochran et renvoie un avertissement dès qu'au moins un effectif théorique est < 5. Si la règle est bien respectée, ne pas tenir compte de l'avertissement.

Une *p-value* significative indique qu'au moins une proportion diffère de sa valeur théorique, sans préciser la(les)quelle(s). Il est dans ce cas nécessaire de tester chaque échantillon/population séparément pour identifier la (les) proportion(s) en question, *via* `prop.multcomp(tab.cont,p=prop.theo)`[1].

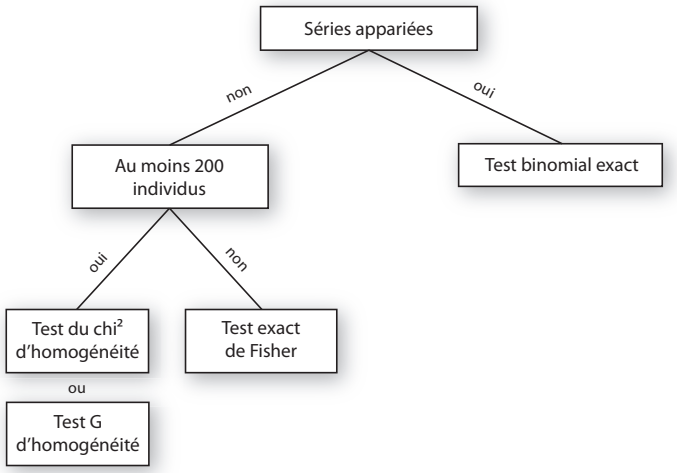
Il peut arriver que les comparaisons multiples n'indiquent aucune différence significative, contrairement au test global. Dans ce cas, la solution la plus prudente est de considérer qu'on ne peut pas savoir quel(le) échantillon/population est responsable du rejet de l'hypothèse nulle dans le test global.

58. Comparaison de deux proportions – 2 catégories

— PACKAGE(S) ADDITIONNEL(S) UTILISÉ(S) DANS CETTE FICHE —

[1]`RVAideMemoire`

Pour choisir le test approprié :



Le test exact de Fisher est toujours le plus fiable mais son temps de calcul augmente avec le nombre d'individus. Lorsque l'effectif est suffisamment grand, l'approximation faite par les tests du χ^2 et G est assez satisfaisante pour qu'ils puissent être utilisés. Les résultats de ces deux tests sont très semblables ; choisir entre l'un et l'autre relève plus d'une habitude que d'une raison statistique.

Séries non appariées

Les données doivent être présentées sous forme d'une matrice à deux lignes et deux colonnes (appelée « tableau de contingence »), où les lignes correspondent aux deux échantillons/populations et les colonnes aux deux catégories :

	Catégorie 1	**Catégorie 2**
Éch/pop 1		
Éch/pop 2		

où sur chaque ligne on a une décompte d'individus dans les deux catégories.
Les proportions testées sont celles de la colonne de gauche (*i.e.* la catégorie 1).

Test exact de Fisher (non paramétrique)

Pour réaliser le test : `fisher.test(tab.cont)` où `tab.cont` est le tableau de contingence.

Test du χ^2 d'homogénéité (non paramétrique)

Pour réaliser le test : `prop.test(tab.cont)`.

Test G d'homogénéité (non paramétrique)

Pour réaliser le test : `G.test(tab.cont)`[1].

Séries appariées

Dans le cas de séries appariées, les individus (au sens statistique) sont reliés par paire. Ce peuvent être des entités réellement différentes, un même individu mesuré deux fois...

Les données doivent toujours être présentées dans un tableau de contingence, mais structuré différemment :

		2nds individus	
		Catégorie 1	Catégorie 2
1ers individus	Catégorie 1		
	Catégorie 2		

où les lignes correspondent aux premiers individus de chaque paire, les colonnes aux seconds.

Test binomial exact (non paramétrique)

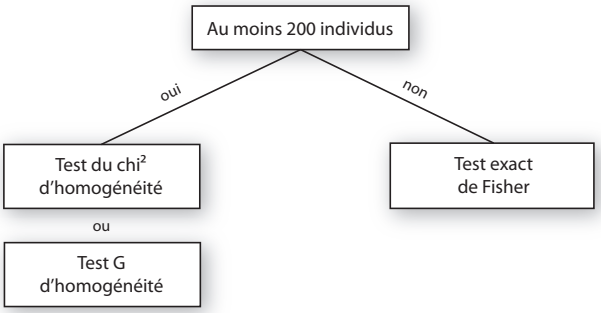
Pour réaliser le test : `binom.test(n1.2,nd)` où `n1.2` est le nombre d'individus dans la case en haut à droite (*i.e.* paires dont le premier individu est de catégorie 1 et le second de catégorie 2), et `nd` est le nombre de paires d'individus dont les valeurs diffèrent (*i.e.* case en bas à gauche + case en haut à droite).

59. Comparaison de plus de deux proportions – 2 catégories

— PACKAGE(S) ADDITIONNEL(S) UTILISÉ(S) DANS CETTE FICHE —

`[1]RVAideMemoire`

Pour choisir le test approprié :



Le test exact de Fisher est toujours le plus fiable mais son temps de calcul augmente avec le nombre d'individus. Lorsque l'effectif est suffisamment grand, l'approximation faite par les tests du χ^2 et G est assez satisfaisante pour qu'ils puissent être utilisés. Les résultats de ces deux tests sont très semblables ; choisir entre l'un et l'autre relève plus d'une habitude que d'une raison statistique.

Les données doivent être présentées sous forme d'une matrice à deux colonnes (appelée « tableau de contingence »), où les lignes correspondent aux échantillons/populations à comparer et les colonnes aux deux catégories :

	Catégorie 1	**Catégorie 2**
Éch/pop 1		
...		
Éch/pop *k*		

où sur chaque ligne on a une décompte d'individus dans les deux catégories.
Les proportions testées sont celles de la colonne de gauche (*i.e.* la catégorie 1).

Test exact de Fisher (non paramétrique)

Pour réaliser le test : `fisher.test(tab.cont)` où `tab.cont` est le tableau de contingence.
Une *p-value* significative indique qu'au moins deux proportions diffèrent l'une de l'autre, sans préciser lesquelles. Il est dans ce cas nécessaire de réaliser des comparaisons deux-à-deux pour identifier les proportions en question, *via* `fisher.multcomp(tab.cont)`[1].
Il peut arriver que les comparaisons deux-à-deux n'indiquent aucune différence significative, contrairement au test global. Dans ce cas, la solution la plus prudente est de considérer qu'on ne peut pas savoir quelles proportions sont responsables du rejet de l'hypothèse nulle dans le test global.

Test du χ^2 d'homogénéité (non paramétrique)

Pour réaliser le test : `prop.test(tab.cont)`.
Si nécessaire, les comparaisons deux-à-deux sont réalisées *via* `pairwise.prop.test(tab.cont)`.

Test G d'homogénéité (non paramétrique)

Pour réaliser le test : `G.test(tab.cont)`[1].
Si nécessaire, les comparaisons deux-à-deux sont réalisées *via* `pairwise.G.test(tab.cont)`[1].

60. Analyser un décompte d'individus dans 2 catégories

— PACKAGE(S) ADDITIONNEL(S) UTILISÉ(S) DANS CETTE FICHE —

[1]`lme4`, [2]`RVAideMemoire`, [3]`MuMIn`, [4]`car`

⚠ L'analyse présentée dans cette fiche est basée sur un modèle. Il est indispensable de maîtriser les notions des fiches **39** à **42**. ⚠

Réponse

Avec une réponse sous forme d'un décompte d'individus dans deux catégories, l'une des deux catégorie est définie comme « catégorie d'intérêt ». Ce qui est analysé est la *proportion représentée par cette catégorie d'intérêt*, appelée « proportion » dans cette fiche.

La réponse doit être une matrice à deux colonnes du type :

```
     Categorie1 Categorie2
[1,]         10         16
[2,]          8         24
[3,]         14         19
...
```

Dans cette matrice, chaque ligne correspond à une répétition et chaque colonne à une catégorie. Pour chaque répétition on a donc un décompte d'individus dans les deux catégories. La colonne de gauche (*i.e.* la catégorie 1) définit la catégorie d'intérêt.

Remarque : au sens statistique, un individu est représenté par une *ligne* du tableau. Pour chaque individu statistique on a donc une proportion (`Categorie1/(Categorie1+Categorie2)`), mais aussi l'*effectif total sur lequel cette proportion est calculée* (`Categorie1+Categorie2`), qui donne plus ou moins de poids aux différents individus (car plus une proportion est calculée sur un effectif élevé, plus elle est précise). L'effectif total peut très bien varier d'une ligne à l'autre.

La matrice réponse peut être obtenue *via* `reponse<-cbind(Categorie1,Categorie2)` où `Categorie1` et `Categorie2` sont des vecteurs correspondant aux deux colonnes (la 1ère valeur correspondant à la 1ère ligne de la matrice et les deux vecteurs étant dans le même ordre). Dans la formule du modèle, la variable à expliquer est cette matrice `reponse`.

Modèle utilisé

Si le principe de l'analyse est le même qu'il y ait des séries appariées ou non, quelques modalités diffèrent. La première est que sans séries appariées le modèle utilisé est un Modèle Linéaire Généralisé (*Generalized Linear Model* ou GLM), tandis qu'avec des séries appariées c'est un Modèle Linéaire Généralisé Mixte (*Generalized Linear Mixed Model* ou GLMM ; « mixte » sous-entendant « contenant au moins un facteur aléatoire », facteurs que l'on utilise précisément pour identifier les séries appariées (voir fiche **11**)).

Contrairement au Modèle Linéaire (et sa variante Mixte, voire fiche **76**), les GLM(M)s ne sont pas forcément basés sur une loi normale. Dans le cas de proportions, la loi à utiliser *a priori* est une loi binomiale.

Remarque : il existe en fait plusieurs types de GLM(M)s basés sur une loi binomiale. Le cas décrit ici – le plus fréquent – est celui d'un modèle *logistique*. Lorsque toutes les variables explicatives sont des covariables, on est dans le cas particulier d'une *régression logistique*.

Construction du modèle

Pour créer le modèle :

— sans séries appariées : `modele<-glm(formule,family="binomial")`

— avec des séries appariées : `modele<-glmer(formule,family="binomial")`[1].

Voir fiche **40** pour une explication détaillée de la construction d'une formule. De manière générale, on peut dire que :

— inclure un facteur permet de tester si les proportions diffèrent entre les niveaux de ce facteur.
— inclure une covariable permet de tester s'il existe une relation entre cette covariable et la variable à expliquer.
— inclure une interaction entre deux variables explicatives permet de tester si l'effet de l'une dépend de l'autre (le raisonnement est le même pour une interaction incluant plus de deux termes). Le cas particulier d'une interaction entre une covariable et un facteur permet de tester si la relation entre la covariable et la variable à expliquer est différente selon la modalité du facteur.

Remarque 1 : Les individus statistiques étant les lignes de la matrice réponse, il y a autant de valeurs par variable explicative que de lignes dans cette matrice.

Remarque 2 : dans le cas où au moins une des variables explicatives est un facteur, et que toutes les proportions d'au moins une modalité sont égales à 0 ou 1, l'ajustement du modèle ne peut pas se faire correctement. Dans ce cas, remplacer `family="binomial"` par `family=binomial(link=elogis()²)`. Trois avertissements sont alors renvoyés à la création du modèle, ne pas en tenir compte. Cette astuce n'est cependant applicable qu'aux GLMs, pas aux GLMMs. Si le modèle est un GLMM, on ne peut simplement pas intégrer proprement dans l'analyse la modalité pour laquelle toutes les valeurs de la variable à expliquer sont identiques.

Vérification de l'ajustement aux données

Avant d'aller plus loin, il est indispensable de vérifier que le modèle s'ajuste bien aux données. Cette étape est fondamentale, et ce pour tout modèle, car un test basé sur un modèle mal ajusté n'est simplement pas fiable (ni valide). Deux conditions sont à vérifier (dans cet ordre).

1. Indépendance entre les résidus du modèle et les valeurs prédites

Voir fiche **41** pour une explication détaillée et la manière de vérifier cette condition. Si elle n'est pas respectée de façon flagrante, plusieurs solutions sont possibles (à tester dans cet ordre) :

1. Changer la *fonction de lien* sur laquelle est basé le modèle. Par défaut cette fonction de lien est le logit et n'est pas précisée (`family="binomial"` est un raccourci pour `family=binomial(link="logit")`). Tester les fonctions de lien probit (`family=binomial(link="probit")`), cauchit (`family=binomial(link="cauchit")`), log (`family=binomial(link="log")`) ou *complementary log-log* (`family=binomial(link="cloglog")`), *i.e.* les quatre autres acceptées par la loi binomiale.
   Remarque 1 : dans le cas de l'utilisation de la fonction `elogis()`², la fonction de lien est appelée « logit empirique ». Cette fonction de lien est difficile à remplacer par une autre. Si l'ajustement du modèle n'est pas bon avec cette fonction de lien, on ne peut simplement pas intégrer proprement dans l'analyse la modalité pour laquelle toutes les valeurs de la variable à expliquer sont identiques.
   Remarque 2 : dans tous les cas, le modèle n'est plus un modèle logistique.
2. Si certaines variables *explicatives* sont quantitatives, en transformer une ou plusieurs ($\sqrt{x}$ et $log(x)$ sont les transformations les plus fréquentes).

2. Absence de sur-dispersion des résidus

Voir fiche **41** pour une explication détaillée. Pour vérifier cette condition, il suffit de calculer le rapport entre la *déviance résiduelle* (*residual deviance*) du modèle et ses *degrés de liberté résiduels* (*residual degrees of freedom*). Si le rapport est > 1.5, il y a sur-dispersion. Pour obtenir ces valeurs :

— GLM : appeler `summary(modele)` et repérer la ligne `Residual deviance: xx.xx on xx degrees of freedom`
— GLMM : utiliser `overdisp.glmer(modele)`².

S'il y a sur-dispersion, il est nécessaire de modifier le modèle :

— GLM : la méthode consiste à modifier la loi sur laquelle est basé le modèle, en remplaçant la loi binomiale par une loi quasi-binomiale *via* `family="quasibinomial"`. La fonction de lien par défaut est le logit, si elle a été modifiée à l'étape précédente il faut garder celle qui a été sélectionnée (par exemple `family=quasibinomial(link="probit")`).
— GLMM : la méthode consiste à ajouter un facteur aléatoire au modèle, dont chaque niveau correspond à un individu (statistique). Cela se fait en deux étapes :

1. Créer le facteur aléatoire : `obs<-factor(1:nrow(reponse))`.
2. Recréer le modèle en ajoutant à la fin de la formule `+(1|obs)`.

Capacité explicative globale

On peut estimer la capacité explicative globale d'un modèle grâce au *coefficient de détermination* (R^2), qui représente la proportion de la variance de la variable à expliquer qui est expliquée par les variables explicatives. Ce coefficient est obtenu *via* `r.squaredGLMM(modele)`[3] (ligne `delta`). La fonction renvoie en fait deux valeurs : le R^2 *marginal* (`R2m`) qui correspond à la part de la variance expliquée uniquement par les facteurs fixes et covariables, et le R^2 *conditionnel* (`R2c`) qui correspond à la part de la variance expliquée par l'ensemble des variables explicatives (fixes et aléatoires). Dans le cas d'un GLM les deux valeurs sont identiques puisqu'il n'y a pas de facteur aléatoire.

Test(s)

Pour tous les modèles évoqués à l'exception de ceux avec une loi quasi-binomiale, l'effet des variables explicatives est testé par la même fonction : `Anova(modele)`[4] (voir fiche **42** pour une explication détaillée des hypothèses testées). Cependant, ce ne sont pas les mêmes tests qui sont réalisés selon le modèle :

— GLM : la fonction réalise un test du rapport des vraisemblances (*Likelihood Ratio Test* ou LR Test) – en fait un test par terme du modèle (*i.e.* un par ligne du tableau renvoyé).

— GLMM : la fonction réalise un test de Wald – en fait un test par terme du modèle (*i.e.* un par ligne du tableau renvoyé).

Dans le cas d'un modèle avec une loi quasi-binomiale, le test à utiliser est un test F. Il est réalisé *via* `Anova(modele,test="F")`[4].

Si un facteur (ou une interaction impliquant un facteur) a un effet significatif, il est nécessaire de réaliser des comparaisons multiples pour identifier les modalités (ou combinaisons de modalités) qui diffèrent. Voir fiche **43** pour réaliser ces comparaisons.

Si une covariable a un effet significatif, la direction de son effet est donnée par le *signe* du paramètre associé. Les valeurs de tous les paramètres du modèle sont obtenues *via* `summary(modele)`. Elles sont appelées `Estimate` et se trouvent dans le tableau `Coefficients` pour un GLM, `Fixed effects` pour un GLMM. Si le coefficient portant le nom de la covariable est négatif, la proportion diminue quand la valeur de la covariable augmente ; s'il est positif, la proportion augmente quand la valeur de la covariable augmente.

Prédiction à partir du modèle

L'intérêt des modèles est à la fois de comprendre l'importance des différentes variables explicatives sur la variable à expliquer, mais également de prédire la valeur que prendrait cette variable à expliquer pour des valeurs *connues* des variables explicatives. Prédire une proportion nécessite donc de fixer la valeur de *toutes* les variables explicatives.

Deux méthodes peuvent être utilisées pour la prédiction (seule la seconde est disponible pour les GLMMs), les deux étant basées sur la fonction `predict()` :

— donner la valeur (ou les valeurs pour faire plusieurs prédictions) de chacune des variables explicatives directement dans la fonction, sous la forme d'une liste : `predict(modele,newdata=list(variables),type="response")`, où `variables` est un enchaînement de `variable1=valeur,variable2=valeur...`

— créer un tableau contenant une colonne par variable explicative (les noms de colonnes doivent être *strictement identiques* aux noms des variables du modèle), et remplir chaque ligne en fonction des valeurs pour lesquelles la prédiction doit être faite (il y a donc une prédiction par ligne de ce tableau). Puis : `predict(modele,newdata=tableau,type="response")`.
Remarque : pour les GLMMs, les valeurs du (ou des) facteur(s) aléatoire(s) peuvent être données ou non. Si ce n'est pas le cas, il est nécessaire d'ajouter l'argument `re.form=NA` à la fonction `predict()`. Les prédictions prennent alors en compte l'effet moyen de toutes les modalités du (ou des) facteur(s) aléatoire(s) du modèle.

EXEMPLE(S)

Avec un modèle contenant un facteur à deux niveaux (A et B), une covariable variant de 0 à 30, et leur interaction :

```
> modele <- glm(reponse~facteur*covariable,family="binomial")
```

On peut prédire une proportion de cette façon :

```
> predict(modele,newdata=list(facteur="A",covariable=10),type="response")
```

Ou, pour plusieurs prédictions :

```
> predict(modele,newdata=list(facteur=c("A","B"),covariable=c(10,10)),type="re-
sponse")
```

Ou encore :

```
> predict(modele,newdata=list(facteur=c("A","B"),covariable=rep(10,2)),type="r-
esponse")
```

Ou encore créer un tableau de ce type :

```
> tableau
  facteur covariable
1       A         10
2       B         10
```

Puis :

```
> predict(modele,newdata=tableau,type="response")
```

Graphes

Effet d'un facteur

Pour illustrer l'effet d'un facteur, on réalise généralement un diagramme en barres où sont représentées les proportions moyennes par modalité. Deux types de moyennes peuvent être représentées :

— les moyennes brutes (*i.e.* calculées à partir des données brutes), avec leurs erreurs standards : voir fiches **35** et **37**, et l'utilisation de la fonction `tapply()`. Attention, ces erreurs standards ne prennent pas en compte le fait que l'effectif total peut varier d'un individu statistique à l'autre. De plus, dans le cas des GLMMs, les moyennes et erreurs standards ne tiennent pas compte du (des) facteur(s) aléatoire(s).
Remarque : la fonction `tapply()` fonctionne à partir d'un vecteur de valeurs numériques. Ici la variable à expliquer est une matrice à deux colonnes, ce qui n'est pas adapté. Il faut donc préalablement calculer manuellement les proportions : `proportions<-Categorie1/(Categorie1+Categorie2)`. C'est sur ce vecteur `proportions` que doit être utilisée la fonction `tapply()`.

— les moyennes ajustées en fonction des autres variables du modèle, également avec leurs erreurs standards : voir fiche **43**.

Une fois les moyennes et erreurs standards récupérées, le diagramme peut être tracé (voir fiche **34**).

Relation avec une covariable

Illustrer la relation entre la variable à expliquer et une covariable nécessite trois étapes :

1. Tracer les points observés : `plot(proportions~covariable)`.
2. Créer un vecteur ayant les mêmes minimum et maximum que la covariable mais découpé en très petits intervalles : `x <- seq2(covariable)`[2].
3. Ajouter la courbe de la relation sur le graphe, ajustée par le modèle. La détermination de tous les points de la courbe est en fait basée sur une prédiction : la valeur que prend la variable à expliquer pour chaque valeur du vecteur `x`. C'est le grand nombre de valeurs de ce vecteur, et le très petit intervalle qui les sépare, qui donne l'aspect lissé de la courbe (qui est en fait constituée de centaines de points reliés entre eux). Si le modèle contient d'autres variables explicatives que la covariable du graphe, il faut en fixer toutes les valeurs à l'identique (de sorte que seule la covariable du graphe change de valeur pour toutes les prédictions). La courbe s'ajoute *via* `lines(x,predict(modele,newdata=variables.à.expliquer,type="response"))`.
Remarque : la fonction `lines()` peut être utilisée plusieurs fois de suite pour tracer plusieurs courbes sur le même graphe, par exemple pour plusieurs niveaux d'un même facteur contenu dans le modèle. Pour modifier le type de tracé et la couleur des courbes, utiliser les arguments `lty` et `col` respectivement (pour plus d'informations et (beaucoup) plus d'options graphiques, voir `?par`). Enfin, pour ajouter une légende, utiliser la fonction `legend()` (voir `?legend` pour plus

d'informations).

EXEMPLE(S)

On se base toujours sur le modèle suivant :

```
> modele <- glm(reponse~facteur*covariable,family="binomial")
```

Étape 1 : tracer les points correspondant aux données observées :

```
> plot(proportions~covariable)
```

Étape 2 : créer le vecteur `x` :

```
> x <- seq2(covariable)
```

[2]

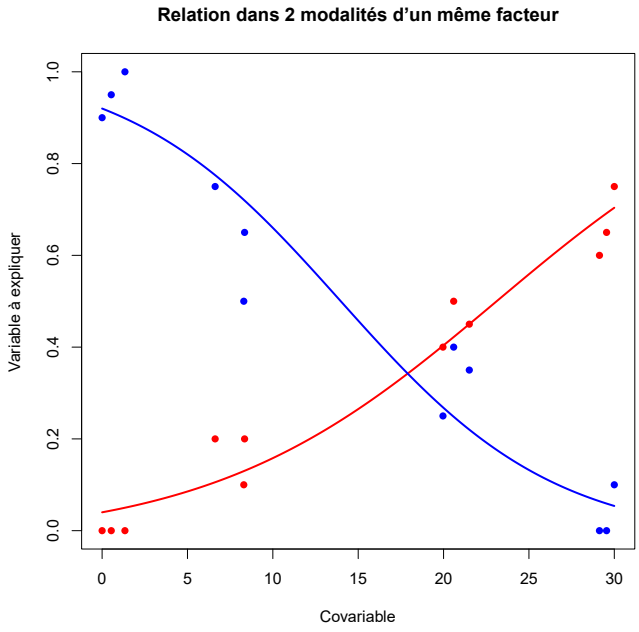
Étape 3 : ajouter la courbe. On choisit de se placer dans la modalité A du facteur :

```
> lines(x,predict(modele,newdata=list(covariable=x,facteur=rep("A",length(x))),
type="response"))
```

La fonction `rep()` est très utile dans cette situation, puisqu'elle permet de créer très simplement un vecteur de la même longueur que `x`, contenant une seule valeur répétée.

Pour ajouter la courbe de la relation dans la modalité B :

```
> lines(x,predict(modele,newdata=list(covariable=x,facteur=rep("B",length(x))),
type="response"))
```



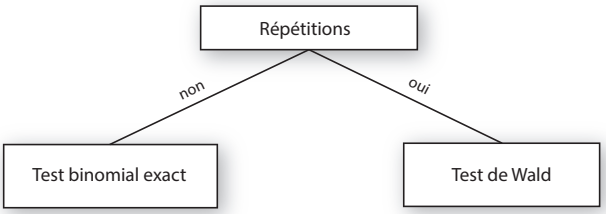
61. Conformité de plusieurs proportions à des valeurs théoriques – plus de 2 catégories

— PACKAGE(S) ADDITIONNEL(S) UTILISÉ(S) DANS CETTE FICHE —

`[1]RVAideMemoire`

La réponse doit être une matrice où chaque ligne correspond à une répétition (il peut n'y en avoir qu'une, dans ce cas la réponse est un vecteur) et chaque colonne à une catégorie. Pour chaque répétition on a donc un décompte d'individus dans chacune des catégories. Les proportions testées sont celles définies par chaque catégorie.

Pour choisir le test approprié :



Test binomial exact (non paramétrique)

Pour réaliser le test : `multinomial.theo.multcomp(reponse,p=prop.theo,prop=TRUE)`[1] où `reponse` est un vecteur à K valeurs (pour K catégories) et `prop.theo` un vecteur donnant la proportion théorique de chaque catégorie (de 1 à K). La somme de ces proportions doit valoir 1. Il y a en fait un test binomial exact par proportion testée.

Test de Wald (paramétrique)

Pour réaliser le test : `wald.ptheo.multinom.test(reponse,p=prop.theo)`[1] où `reponse` est une matrice à K colonnes (pour K catégories) et autant de lignes qu'il y a de répétitions. Il y a en fait un test de Wald par proportion testée.

Remarque : les proportions calculées tiennent compte du fait que l'effectif total de chaque échantillon peut varier.

— EXEMPLE(S) —

Les résultats sont les suivants :

```
> reponse
     Categorie1 Categorie2 Categorie3
[1,]         10         16         16
[2,]          8         24         17
[3,]         14         19         20
```

On compare les proportions observées aux proportions théoriques 0.25 / 0.5 / 0.25 :

```
> wald.ptheo.multinom.test(reponse,p=c(0.25,0.5,0.25))[1]
```

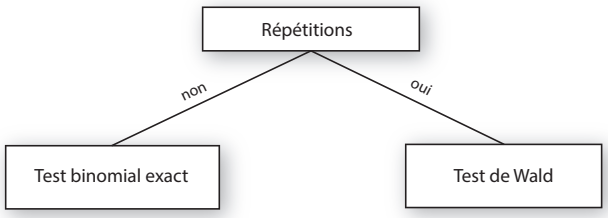
62. Comparaison de plusieurs proportions – plus de 2 catégories

PACKAGE(S) ADDITIONNEL(S) UTILISÉ(S) DANS CETTE FICHE

[1]RVAideMemoire

La réponse doit être une matrice où chaque ligne correspond à une répétition (il peut n'y en avoir qu'une, dans ce cas la réponse est un vecteur) et chaque colonne à une catégorie. Pour chaque répétition on a donc un décompte d'individus dans chacune des catégories. Les proportions comparées sont celles définies par chaque catégorie.

Pour choisir le test approprié :



Test binomial exact (non paramétrique)

Pour réaliser le test : `multinomial.multcomp(reponse)`[1] où `reponse` est un vecteur à K valeurs (pour K catégories). La fonction réalise en fait directement toutes les comparaisons multiples entre les différentes catégories.

Test de Wald (paramétrique)

Pour réaliser le test : `prop.multinom.test(reponse)`[1] où `reponse` est une matrice à K colonnes (pour K catégories) et autant de lignes qu'il y a de répétitions. La fonction réalise en fait directement toutes les comparaisons multiples entre les différentes catégories.

EXEMPLE(S)

Les résultats sont les suivants :

```
> reponse
     Categorie1 Categorie2 Categorie3
[1,]         10         16         16
[2,]          8         24         17
[3,]         14         19         13
```

La proportion de chaque catégorie (et son erreur standard) est donnée par :

```
> prop.multinom(reponse)[1]
$probs
Categorie1 Categorie2 Categorie3
    0.2336     0.4307     0.3358

$se
Categorie1 Categorie2 Categorie3
    0.0416     0.0325     0.0285
```

On compare ces proportions :

```
> prop.multinom.test(reponse)[1]
```

63. Analyser un décompte d'individus dans plus de 2 catégories

— PACKAGE(S) ADDITIONNEL(S) UTILISÉ(S) DANS CETTE FICHE —

[1]`nnet`, [2]`RVAideMemoire`, [3]`car`

⚠ L'analyse présentée dans cette fiche est basée sur un modèle. ⚠
Il est indispensable de maîtriser les notions des fiches **39** à **42**.

Réponse

Avec une réponse sous forme d'un décompte d'individus dans plus de deux catégories, ce qui est analysé est la *proportion représentée par chaque catégorie*. Il sagit donc d'une réponse multiple, et non unique comme dans la plupart des situations classiques (comme l'analyse d'un décompte d'individus dans deux catégories, voir fiche **60**). Cela complique sérieusement l'interprétation des résultats.

Pour pouvoir analyser une telle réponse, il est nécessaire de définir une « catégorie de référence ». Car ce sur ce quoi va travailler le modèle est en fait *le rapport entre la proportion d'une catégorie et celle de la catégorie de référence*. Pour K catégories, on doit donc interpréter $K-1$ rapports par variable explicative, qui s'étendent à $\frac{K.(K-1)}{2}$ rapports (par variable explicative !) pour toutes les combinaisons deux-à-deux des K catégories. On voit bien la complexité de travailler sur de telles réponses. D'où l'idée de se restreindre à un design expérimental très simple (*i.e.* peu de variables explicatives et encore moins d'interactions) et à un nombre de catégories (*i.e.* K) limité. Pour des raisons analogues, mieux vaut éviter les variables explicatives qualitatives à plus de deux niveaux (ou alors il faut être prêt à s'accrocher pour l'interprétation, qui peut être fastidieuse).

Dans **R**, la réponse doit être une matrice à K colonnes du type :

```
     Categorie1 Categorie2 Categorie3
[1,]         10         16         16
[2,]          8         24         17
[3,]         14         19         13
 ...
```

Dans cette matrice, chaque ligne correspond à une répétition et chaque colonne à une catégorie. Pour chaque répétition on a donc un décompte d'individus dans les K catégories. La colonne de gauche (*i.e.* la catégorie 1) définit la catégorie d'intérêt. Le choix de cette catégorie n'a aucune incidence sur l'analyse.

Remarque : au sens statistique, un individu est représenté par une *ligne* du tableau. Pour chaque individu statistique on a donc une série de proportions, mais aussi l'*effectif total sur lequel ces proportions sont calculées*, qui donne plus ou moins de poids aux différents individus (car plus une proportion est calculée sur un effectif élevé, plus elle est précise). L'effectif total peut très bien varier d'une ligne à l'autre.

La matrice réponse peut être obtenue *via* `reponse<-cbind(Categorie1,Categorie2,Categorie3)` où `Categorie1`, `Categorie2` et `Categorie3` sont des vecteurs correspondant aux trois colonnes (la 1ère valeur correspondant à la 1ère ligne de la matrice et les trois vecteurs étant dans le même ordre). Dans la formule du modèle, la variable à expliquer est cette matrice `reponse`.

Modèle utilisé

Le modèle utilisé est un modèle multinomial (ou « polytomique non ordonné »), une extension du Modèle Linéaire Généralisé (*Generalized Linear Model* ou GLM) pour décompte d'individus dans deux catégories (voir fiche **60**).

Remarque : il existe en fait plusieurs types de modèles multinomiaux. Le cas décrit ici – le plus fréquent – est celui d'un modèle *logistique*. Lorsque toutes les variables explicatives sont des covariables, on est dans le cas particulier d'une *régression multinomiale logistique*.

Construction du modèle

Pour créer le modèle : `modele<-multinom(formule,abstol=1e-15,reltol=1e-15,maxit=1000)`[1]. Voir fiche **40** pour une explication détaillée de la construction d'une formule. Les arguments `abstol`, `reltol` et `maxit` sont des options assurant une plus grande précision des résultats.

Remarque : la construction d'un modèle multinomial par la fonction `multinom()`[1] est un processus itératif (comme très souvent) qui a la particularité d'être en partie visible. Pour supprimer les messages affichés à la construction du modèle, ajouter l'argument `trace=FALSE`.

Vérification de la dépendance des paramètres du modèle aux données

La variable à expliquer n'étant pas quantitative, la vérification de la validité du modèle ne se fait pas de manière habituelle ici. On cherche en fait à savoir à quel point les paramètres du modèle dépendent des données qu'on lui fournit. Si les paramètres sont très sensibles à la moindre variation dans les données, c'est que le modèle est mal défini. Cela peut indiquer (entre autres) que le modèle peut être simplifié ou que certains paramètres ne sont pas calculables.

La valeur qui représente cette dépendance est le *conditionnement* de la matrice hessienne. Elle est obtenue *via* `cond.multinom(modele)`[2]. Il n'y a pas vraiment de seuil absolu, mais on considère généralement que si elle est supérieure à 10^5 c'est que le modèle est mal défini.

Test(s)

L'effet des variables explicatives est testé *via* `Anova(modele)`[3] (voir fiche **42** pour une explication détaillée des hypothèses testées). Le test réalisé est un test du rapport des vraisemblances (*Likelihood Ratio Test* ou LR Test) – en fait un test par terme du modèle (*i.e.* un par ligne du tableau renvoyé).

Si une variable explicative a un effet significatif, cela indique qu'elle influence *le rapport entre au moins deux catégories* de la variable réponse. Autrement dit, la proportion d'une catégorie donnée *par rapport à celle d'une autre catégorie donnée*. Pour identifier quel(s) rapport(s) est (sont) significativement influencé(s) par la variable explicative en question : `test.multinom(modele,variable)`[2] où `variable` est la variable explicative dont on souhaite étudier l'influence en détail. Ce qui est renvoyé par `test.multinom()`[2] dépend de la nature de la variable explicative :

— Pour une variable explicative quantitative : la fonction renvoie un tableau où chaque ligne correspond à un rapport entre deux catégories de la variable réponse (la syntaxe `A|B` signifie « proportion de A par rapport à la proportion de B »). Le *signe* du coefficient indique le sens de la relation entre un rapport et la variable explicative. L'*odds ratio* est une façon simple de comprendre ce rapport, qui indique de combien il varie *pour une augmentation d'une unité de la variable explicative*.
Remarque : il n'est pas surprenant que les *p-values* des tests par rapport soient moins nettes que la *p-value* globale obtenue *via* `Anova(modele)`[3]. D'abord parce qu'un test global ne se réduit pas à une somme de tests individuels, mais aussi car ce ne sont pas les mêmes tests qui sont réalisés : test du rapport des vraisemblances au niveau global, tests de Wald pour chaque rapport. Le second est moins puissant que le premier (voir fiche **17**).

EXEMPLE(S)

Sur un modèle dont la réponse a trois catégories (A/B/C), une covariable a un effet significatif. Le détail de son influence est le suivant :

```
       Coeff      SE Odds.ratio       z Pr(>|z|)
A|C -0.93509 0.57383     0.3926 -1.6296  0.10319
B|C  1.56595 1.05528     4.7872  1.4839  0.13783
B|A  2.50104 1.20247    12.1952  2.0799  0.03753 *
--
Signif. codes: 0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

La conclusion est que la covariable n'influence que le rapport entre les catégories A et B (seule *p-value* significative). D'après l'odds ratio, la proportion de la catégorie B est 12.2 fois plus importante que celle de la catégorie A quand la covariable augmente d'une unité.
Remarque : un coefficient significativement différent de 0 revient à un odds ratio significativement différent de 1.

— Pour une variable explicative qualitative à deux niveaux : la fonction renvoie un tableau structuré comme pour une covariable. L'odds ratio d'un rapport entre deux catégories de la variable réponse indique le rapport entre (i) ce rapport dans une classe de la variable explicative et (ii) ce même rapport dans l'autre classe (c'est donc un rapport de rapports). La direction du rapport entre les classes de la variable explicative est indiquée en titre du tableau.

EXEMPLE(S)

Sur un modèle dont la réponse a trois catégories (A/B/C), un facteur à deux niveaux (Femelle/Mâle) a un effet significatif. Le détail de son influence est le suivant :

```
$`Male|Femelle`
      Coeff     SE Odds.ratio        z Pr(>|z|)
A|C  0.1702 1.8533    1.18550  0.09182  0.92684
B|C -1.9660 0.8645    0.14002 -2.27415  0.02296 *
B|A -5.1361 4.0226    0.00588 -1.27682  0.20166
--
Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

La conclusion est que le facteur n'influence que le rapport entre les catégories B et C. D'après l'odds ratio, la proportion de B par rapport à C chez les mâles est seulement 0.14 fois celle des femelles (autrement dit chez les femelles la proportion de B par rapport à C est $\frac{1}{0.14} = 7.14$ fois plus grande que chez les mâles).

— Pour une variable explicative qualitative à plus de deux niveaux : la fonction renvoie un tableau *par rapport possible entre les classes de la variable explicative*. C'est donc une extension du cas précédent, forcément bien plus difficile à interpréter.

Prédiction à partir du modèle

L'intérêt des modèles est à la fois de comprendre l'importance des différentes variables explicatives sur la variable à expliquer, mais également de prédire la valeur que prendrait cette variable à expliquer pour des valeurs *connues* des variables explicatives. Prédire une série de proportions nécessite donc de fixer la valeur de *toutes* les variables explicatives.

Deux méthodes peuvent être utilisées pour la prédiction, les deux étant basées sur la fonction `predict()` :

— donner la valeur (ou les valeurs pour faire plusieurs prédictions) de chacune des variables explicatives directement dans la fonction, sous la forme d'une liste : `predict(modele,newdata=list(variables,type="probs"))`, où `variables` est un enchaînement de `variable1=valeur, variable2=valeur...`

— créer un tableau contenant une colonne par variable explicative (les noms de colonnes doivent être *strictement identiques* aux noms des variables du modèle), et remplir chaque ligne en fonction des valeurs pour lesquelles la prédiction doit être faite (il y a donc une prédiction par ligne de ce tableau). Puis : `predict(modele,newdata=tableau,type="probs")`.

La fonction `predict()` renvoie, pour chaque prédiction, la proportion de chacune des catégories de la variable réponse. Le résultat est une matrice où chaque ligne est une prédiction et chaque colonne une catégorie.

EXEMPLE(S)

Avec un modèle contenant un facteur à deux niveaux (A et B) et une covariable variant de 0 à 30 :

```
> modele <- multinom(reponse~facteur+covariable,abstol=1e-15,reltol=1e-15,maxit=1000)[1]
```

On peut prédire la classe la plus probable de cette façon :

```
> predict(modele,newdata=list(facteur="A",covariable=10),type="probs")
```

Ou, pour plusieurs prédictions :

```
> predict(modele,newdata=list(facteur=c("A","B"),covariable=c(10,10)),type="probs")
```

Ou encore :

```
> predict(modele,newdata=list(facteur=c("A","B"),covariable=rep(10,2)),type="p-
robs")
```

Ou encore créer un tableau de ce type :

```
> tableau
  facteur covariable
1       A         10
2       B         10
```

Puis :

```
> predict(modele,newdata=tableau,type="probs")
```

64. Analyser une réponse continue bornée

— PACKAGE(S) ADDITIONNEL(S) UTILISÉ(S) DANS CETTE FICHE —

[1]`RVAideMemoire`, [2]`betareg`, [3]`glmmTMB`, [4]`MuMIn`, [5]`car`

⚠ L'analyse présentée dans cette fiche est basée sur un modèle. Il est indispensable de maîtriser les notions des fiches **39** à **42**. ⚠

Réponse

On considèrera ici le cas le plus fréquent, celui où la réponse est une proportion (*i.e.* bornée entre 0 et 1). Si la réponse est bornée dans un autre intervalle, commencer par la ramener dans l'intervalle [0 ; 1] en appliquant la formule suivante : $\frac{x-min_x}{max_x-min_x}$ où min_x et max_x sont les valeurs minimale et maximale *possibles* de la série de données x (même si ces valeurs ne sont pas atteintes). Attention, la série de données doit en réalité être dans l'intervalle]0 ; 1[, *i.e.* elle ne doit contenir aucun 0 ni 1. Si c'est le cas, transformer la réponse *via* : `reponse<-p.beta(x)`[1] où `x` est la série de données x dans l'intervalle [0 ; 1].

Remarque : une variable bornée uniquement à 0 (*i.e.* continue mais dont les valeurs ne peuvent être que nulles ou positives, sans limite supérieure) peut être analysée comme un temps avant la survenue d'un évènement (voire fiche **79**).

— EXEMPLE(S) —

Pour travailler sur une variable `x` initialement bornée entre 1 et 5, on commence par la ramener entre 0 et 1 :

```
> x2 <- (x-1)/(5-1)
```

Si `x` contenait des 1 ou des 5 (donc si `x2` contient des 0 ou des 1), une seconde étape est nécessaire pour revenir dans l'intervalle]0 ; 1[:

```
> reponse <- p.beta(x2)[1]
```

Modèle utilisé

Si le principe de l'analyse est le même qu'il y ait des séries appariées ou non, quelques modalités diffèrent. La première est que sans séries appariées le modèle utilisé est une régression bêta, tandis qu'avec des séries appariées c'est une régression bêta mixte ; « mixte » sous-entendant « contenant au moins un facteur aléatoire », facteurs que l'on utilise précisément pour identifier les séries appariées (voir fiche **11**). Contrairement au Modèle Linéaire (et sa variante Mixte, voire fiche **76**), la régression bêta n'est pas basée sur une loi normale mais… sur une loi bêta.

Remarque : la régression bêta a beaucoup d'affinités avec le Modèle Linéaire Généralisé (GLM), utilisé pour analyser d'autres types de réponse (voir fiches **48**, **55** et **60** par exemple). Pour des raisons très techniques, la régression bêta n'est cependant pas un cas particulier de GLM.

Construction du modèle

Pour créer le modèle :

— sans séries appariées : `modele<-betareg(formule)`[2]
— avec des séries appariées : `modele<-glmmTMB(formule,family=beta_family)`[3].

Voir fiche **40** pour une explication détaillée de la construction d'une formule. De manière générale, on peut dire que :

— inclure un facteur permet de tester si les proportions diffèrent entre les niveaux de ce facteur.
— inclure une covariable permet de tester s'il existe une relation entre cette covariable et la variable à expliquer.
— inclure une interaction entre deux variables explicatives permet de tester si l'effet de l'une dépend de l'autre (le raisonnement est le même pour une interaction incluant plus de deux termes). Le cas particulier d'une interaction entre une covariable et un facteur permet de tester si la relation entre

la covariable et la variable à expliquer est différente selon la modalité du facteur.

Vérification de l'ajustement aux données

Avant d'aller plus loin, il est indispensable de vérifier que le modèle s'ajuste bien aux données. Cette étape est fondamentale, et ce pour tout modèle, car un test basé sur un modèle mal ajusté n'est simplement pas fiable (ni valide). Deux conditions sont à vérifier (dans cet ordre).

1. Indépendance entre les résidus du modèle et les valeurs prédites

Voir fiche **41** pour une explication détaillée et la manière de vérifier cette condition. Si elle n'est pas respectée de façon flagrante, plusieurs solutions sont possibles (à tester dans cet ordre) :

1. Changer la *fonction de lien* sur laquelle est basé le modèle. Par défaut cette fonction de lien est le logit et n'est pas précisée. Tester les fonctions de lien probit, log ou *complementary log-log*, de la manière suivante :
   — Régression bêta : en ajoutant l'argument `link="probit"`, `link="log"` ou `link="cloglog"`.
   — Régression bêta mixte : en remplaçant `family=beta_family` par`family=beta_family(link="probit")`, `family=beta_family(link="log")` ou `family=beta_family(link="cloglog")`.
2. Si certaines variables *explicatives* sont quantitatives, en transformer une ou plusieurs ($\sqrt{x}$ et $log(x)$ sont les transformations les plus fréquentes).

2. Modélisation correcte de la variance des résidus

Voir fiche **41** pour une explication détaillée. Cette condition est vérifiée si, sur le graphe renvoyé par la fonction `plotresid()`[1], la dispersion verticale des points est à peu près constante sur toute la longueur de l'axe des abscisses. Si cette condition n'est pas vérifiée, il est nécessaire de modifier le modèle pour modéliser correctement la variance des résidus. Cette modélisation se fait *en fonction de variables explicatives identifiées*. Celles-ci peuvent être les mêmes, en partie les mêmes, ou pas les mêmes que les variables explicatives déjà incluses dans le modèle. En première approche, si l'on n'a pas trop d'idée de comment modéliser la variance des résidus, on peut simplement reprendre les variables explicatives déjà incluses dans le modèle (mais attention, plus le modèle est complexe et plus il y a des risques que ça ne fonctionne pas bien et que **R** renvoie des erreurs...), quitte à simplifier dans un second temps. La manière de s'y prendre dépend du type de modèle qui est utilisé :

— Régression bêta : la modélisation de la variance des résidus est directement incluse dans la formule du modèle, *via* `modele<-betareg(reponse~form1|form2)`[2], où `form1` est la partie de la formule qui contient les variables explicatives déjà incluses dans le modèle, et `form2` est la partie de la formule qui contient les variables explicatives servant à modéliser la variance des résidus (l'argument `link` pouvant également être ajouté).
— Régression bêta mixte : la modélisation de la variance des résidus se fait dans une formule séparée, en ajoutant l'argument `dispformula=~form2`.

EXEMPLE(S)

Si la variance des résidus n'est pas bien modélisée dans un modèle incluant simplement deux facteurs A et B (sans modélisation particulière de cette variance) :

```
> modele1 <- betareg(reponse~A+B)²
```

On peut modéliser la variance des résidus en utilisant les mêmes variables explicatives :

```
> modele2 <- betareg(reponse~A+B|A+B)²
```

La fonction `plotresid()`[1] aidera à déterminer si les deux variables explicatives sont vraiment nécessaires pour modéliser correctement la variance des résidus, ou s'il est possible de faire plus simple. On peut en effet observer les résidus des modèles suivants :

```
> modele3 <- betareg(reponse~A+B|A)²
> modele4 <- betareg(reponse~A+B|B)²
```

Si l'un de ces deux modèles donne des résultats aussi bons que le modèle 2, on privilégiera alors le plus simple (celui dans lequel la variance des résidus est modélisée par une seule variable explicative).

Si le modèle de départ était un modèle mixte (avec un facteur aléatoire C) :
`> modele1 <- glmmTMB(reponse~A+B+(1|C),family=beta_family)`[3]
On peut modéliser la variance des résidus en ajoutant l'argument `dispformula`. En première approche on peut inclure toutes les variables explicatives (hors facteurs aléatoires) :
`> modele2 <- glmmTMB(reponse~A+B+(1|C),dispformula=~A+B,family=beta_family)`[3]
La même question qu'au-dessus se pose de simplifier la modélisation de la variance des résidus. Cependant, pour des raisons techniques on ne peut pas utiliser le graphe renvoyé par la fonction `plotresid()`[1] pour estimer la variance des résidus quand celle-ci est modélisée explicitement. On peut alors s'appuyer sur une démarche de sélection de modèle, basée sur un critère comme l'AICc (voir fiche **44**). Pour mettre en œuvre cette démarche, commencer par créer les modèles qui modélisent plus simplement la variance des résidus :
`> modele3 <- glmmTMB(reponse~A+B+(1|C),dispformula=~A,family=beta_family)`[3]
`> modele4 <- glmmTMB(reponse~A+B+(1|C),dispformula=~B,family=beta_family)`[3]
Puis calculer l'AICc de tous les modèles dans lesquels la variance des résidus est modélisée :
`> AICc(modele2,modele3,modele4)`[4]
On retiendra le modèle avec la valeur d'AICc la plus *faible* (voir fiche **44**).

Capacité explicative globale

On peut estimer la capacité explicative globale d'un modèle grâce au *coefficient de détermination* (R^2), qui représente la proportion de la variance de la variable à expliquer qui est expliquée par les variables explicatives. Pour une régression bêta, ce coefficient est obtenu *via* `summary(modele)$pseudo.r.squared`. Le calcul du coefficient de détermination n'est malheureusement pas encore disponible pour une régression bêta mixte.

Test(s)

Quel que soit le modèle, l'effet des variables explicatives est testé par un test de Wald – en fait un test par terme du modèle (*i.e.* un par ligne du tableau renvoyé) –, réalisé *via* `Anova(modele)`[5] (voir fiche **42** pour une explication détaillée des hypothèses testées).

Si un facteur (ou une interaction impliquant un facteur) a un effet significatif, il est nécessaire de réaliser des comparaisons multiples pour identifier les modalités (ou combinaisons de modalités) qui diffèrent. Voir fiche **43** pour réaliser ces comparaisons.

Si une covariable a un effet significatif, la direction de son effet est donnée par le *signe* du paramètre associé. Les valeurs de tous les paramètres du modèle sont obtenues *via* `summary(modele)`. Elles sont appelées `Estimate` et se trouvent dans le tableau `Coefficients` pour une régression bêta, dans le tableau `Conditional model` pour une régression bêta mixte. Si le coefficient portant le nom de la covariable est négatif, la proportion diminue quand la valeur de la covariable augmente ; s'il est positif, la proportion augmente quand la valeur de la covariable augmente.

Prédiction à partir du modèle

L'intérêt des modèles est à la fois de comprendre l'importance des différentes variables explicatives sur la variable à expliquer, mais également de prédire la valeur que prendrait cette variable à expliquer pour des valeurs *connues* des variables explicatives. Prédire une proportion nécessite donc de fixer la valeur de *toutes* les variables explicatives.

Pour prédire une proportion, commencer par créer un tableau contenant une colonne par variable explicative (les noms de colonnes doivent être *strictement identiques* aux noms des variables du modèle), et remplir chaque ligne en fonction des valeurs pour lesquelles la prédiction doit être faite (il y a donc une prédiction par ligne de ce tableau). Puis : `predict(modele,newdata=tableau,type="response")`.

Remarque 1 : pour les régressions bêta mixtes, les valeurs du (ou des) facteur(s) aléatoire(s) peuvent être données ou non. Si ce n'est pas le cas, il faut quand même une colonne pour chaque facteur aléatoire dans `tableau` mais ces colonnes sont remplies de `NA`. Les prédictions prennent alors en compte l'effet moyen de toutes les modalités du (ou des) facteur(s) aléatoire(s) du modèle.

Remarque 2 : si la variable réponse a été transformée en amont de la construction du modèle, il faut transformer les prédictions du modèle exactement en sens inverse. L'inverse de la transformation réalisée par `p.beta()`[1] est obtenu *via* : `p.beta(pred,n=nb,back=TRUE)`[1] où `pred` est un vecteur contenant la

(ou les) valeur(s) prédite(s) et `nb` le nombre d'individus sur lequel le modèle a été construit (typiquement le nombre de lignes du jeu de données, mais attention à ne pas compter les individus pour lesquels il y a des données manquantes car ces individus ne sont pas pris en compte par le modèle). L'inverse de la transformation $\frac{x-min_x}{max_x-min_x}$ est obtenu en appliquant la formule suivante : $pred.(max_x - min_x) + min_x$.

EXEMPLE(S)

Un modèle a été construit qui explique une réponse initialement bornée entre 1 et 5 par un facteur à deux niveaux (A et B), une covariable variant de 0 à 30, et leur interaction. Le jeu de données sur lequel ce modèle a été construit contient 30 individus (*i.e.* 30 lignes) et la réponse contenait des valeurs 1 et 5. Le modèle est le suivant :

```
> modele <- betareg(reponse~facteur*covariable)
```
[2]

On peut prédire une proportion en créant un tableau de ce type :

```
> tableau
  facteur covariable
1       A         10
2       B         10
```

Pour otbtenir les prédictions dans l'intervalle]0 ; 1[:

```
> predictions <- predict(modele,newdata=tableau,type="response")
```

On ramène ces prédictions dans l'intervalle [0 ; 1] :

```
> predictions2 <- p.beta(predictions,n=30,back=TRUE)
```
[1]

On ramène finalement ces prédictions dans l'intervalle initial, *i.e.* entre 1 et 5 :

```
> predictions3 <- predictions2*(5-1)+1
```

Graphes

Effet d'un facteur

Pour illustrer l'effet d'un facteur, on réalise généralement un diagramme en barres où sont représentées les proportions moyennes par modalité. Deux types de moyennes peuvent être représentées :

— les moyennes brutes (*i.e.* calculées à partir des données brutes), avec leurs erreurs standards : voir fiches **35** et **37**, et l'utilisation de la fonction `tapply()`. Attention, dans le cas d'une régression bêta mixte ces moyennes et erreurs standards ne tiennent pas compte du (des) facteur(s) aléatoire(s).

— les moyennes ajustées en fonction des autres variables du modèle, également avec leurs erreurs standards : voir fiche **43**.

Une fois les moyennes et erreurs standards récupérées, le diagramme peut être tracé (voir fiche **34**).

Relation avec une covariable

Illustrer la relation entre la variable à expliquer et une covariable nécessite trois étapes :

1. Tracer les points observés : `plot(reponse~covariable)`. Attention, si le modèle a été construit sur des données transormées, `reponse` correspond ici aux données initiales, *non transformées*.
2. Créer un vecteur ayant les mêmes minimum et maximum que la covariable mais découpé en très petits intervalles : `x <- seq2(covariable)`[1].
3. La détermination de tous les points de la courbe est en fait basée sur une prédiction : la valeur que prend la variable à expliquer pour chaque valeur du vecteur `x`. C'est le grand nombre de valeurs de ce vecteur, et le très petit intervalle qui les sépare, qui donne l'aspect lissé de la courbe (qui est en fait constituée de centaines de points reliés entre eux). Récupérer ainsi la valeur prédite par le modèle pour chaque valeur de `x` (attention aux transformations qui peuvent être nécessaires pour revenir à l'intervalle initial). Si le modèle contient d'autres variables explicatives que la covariable du graphe, il faut en fixer toutes les valeurs à l'identique (de sorte que seule la covariable du graphe change de valeur pour toutes les prédictions).
4. Ajouter la courbe de la relation sur le graphe, ajustée par le modèle, *via* `lines(x,pred)` où `pred` est le vecteur contenant les valeurs prédites à l'étape précédente.
   Remarque : la fonction `lines()` peut être utilisée plusieurs fois de suite pour tracer plusieurs courbes sur le même graphe, par exemple pour plusieurs niveaux d'un même facteur contenu dans le modèle. Pour modifier le type de tracé et la couleur des courbes, utiliser les arguments `lty` et `col` respectivement (pour plus d'informations et (beaucoup) plus d'options graphiques, voir `?par`). Enfin, pour ajouter une légende, utiliser la fonction `legend()` (voir `?legend` pour plus d'informations).

EXEMPLE(S)

Pour tracer un graphe à partir du modèle construit dans l'exemple précédent (qui explique une réponse initialement bornée entre 1 et 5, et qui est basé sur 30 individus), la démarche est la suivante.

Étape 1 : tracer les points correspondant aux données *non transformées* observées :

```
> plot(reponse~covariable)
```

Étape 2 : créer le vecteur `x` :

```
> x <- seq2(covariable)
```
[1]

Étape 3 : prédire les valeurs de la réponse pour chaque valeur de `x` (*cf.* exemple précedent). On choisit de se placer dans la modalité A du facteur :

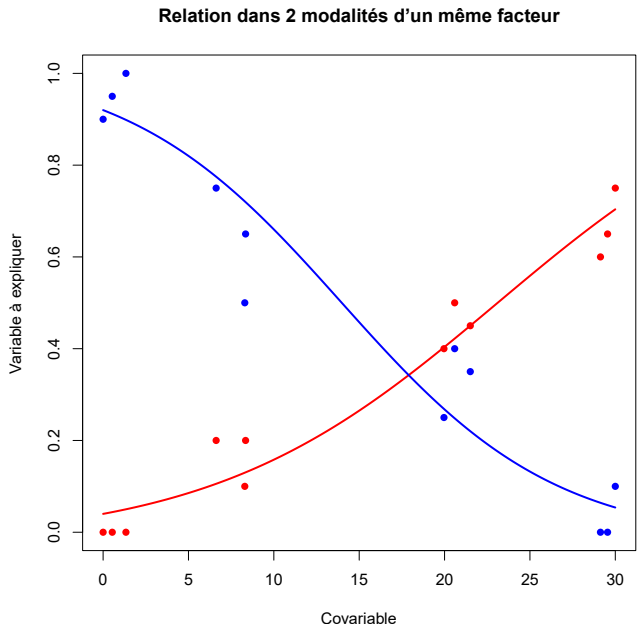
```
> tableau <- data.frame(covariable=x,facteur=rep("A",length(x)),stringsAsFactors=TRUE)
```
(voir fiche **5** ; la fonction `rep()` est très utile dans cette situation, puisqu'elle permet de créer très simplement un vecteur de la même longueur que `x`, contenant une seule valeur répétée)

```
> predictions <- predict(modele,newdata=tableau,type="response")
> predictions2 <- p.beta(predictions,n=30,back=TRUE)
```
[1]
```
> predictions3 <- predictions2*(5-1)+1
```

Étape 4 : ajouter la courbe :

```
> lines(x,predictions3)
```

Pour ajouter la courbe de la relation dans la modalité B, répéter les étapes 3 et 4 en ayant remplacé `"A"` par `"B"` à la création de `tableau`.



65. Conformité d'une variable continue à une distribution théorique

— PACKAGE(S) ADDITIONNEL(S) UTILISÉ(S) DANS CETTE FICHE —

[1]`car`, [2]`RVAideMemoire`, [3]`goftest`

L'ajustement à une distribution théorique se teste avant tout *graphiquement*, car il est rare que des données réelles suivent parfaitement une loi théorique. Il y a donc toujours une part de subjectivité dans l'estimation d'un ajustement, que les tests statistiques ne peuvent pas refléter.

Test graphique

Le test visuel est réalisé grâce au graphe quantile-quantile, tracé de la manière suivante : `qqPlot(serie,dist="loi",par)`[1] où `serie` est un vecteur contenant la série de données, `loi` est la loi théorique choisie (entre guillemets) et `par` ses paramètres séparés par une virgule (voir fiches **23** à **28**).

— EXEMPLE(S) —

Ajustement à une loi exponentielle (voir fiche **25**) :

```
> qqPlot(serie,dist="exp",lambda)[1]
```

Ajustement à une loi de χ^2 (voir fiche **26**) :

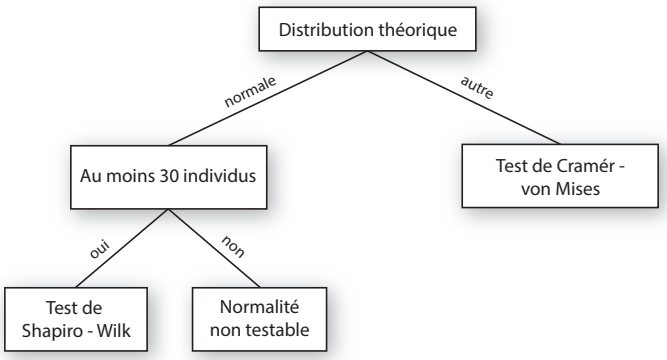
```
> qqPlot(serie,dist="chisq",ddl)[1]
```

Dans le cas d'une loi normale, le même résultat est obtenu directement *via* `qqPlot(serie)`[1], la loi normale étant celle utilisée par défaut. Pour tracer un graphe par niveau d'un facteur : `byf.qqnorm(reponse~facteur)`[2] où `reponse` et `facteur` sont des vecteurs contenant la valeur de chaque individu pour les deux variables (dans le même ordre). Le symbole `~` signifie « expliqué par » ou « en fonction de ». Dans le cas d'un ajustement à une loi normale multivariée, utiliser les fonctions `mqqnorm()`[2] et `byf.mqqnorm()`[2] (voir `?mqqnorm` et `?byf.mqqnorm` pour leur utilisation).

La distribution de la série suit la loi théorique choisie si les points du graphe sont à peu près alignés sur une droite. Toute autre structuration des points (courbure(s), nombreux points éloignés...) indique le contraire. La droite tracée est celle qui passe par les 1er et 3ème quartiles de la distribution représentée, et son intervalle de confiance est affiché en pointillés. Les points peuvent ne pas être parfaitement alignés sur la droite, mais s'ils restent dans l'intervalle de confiance l'ajustement est considéré comme correct.

Test statistique

Pour choisir le test approprié :



Ajustement à une loi normale – Test de Shapiro - Wilk (non paramétrique)

Pour réaliser le test : `shapiro.test(serie)`. Pour tester la normalité de la distribution d'une variable par niveau d'un facteur : `byf.shapiro(reponse~facteur)`[2]. Dans le cas d'un ajustement à une loi normale multivariée, utiliser les fonctions `mshapiro.test()`[2] et `byf.mshapiro()`[2] (voir `?mshapiro.test`

et `?byf.mshapiro` pour leur utilisation).

Ajustement à une loi autre que normale – Test de Cramér - von Mises (non paramétrique)

Pour réaliser le test : `cvm.test(serie,ploi,par)`[3] où `loi` est la loi choisie et `par` ses paramètres séparés par une virgule (voir fiches **23** à **28**).

EXEMPLE(S)

Ajustement à une loi exponentielle (voir fiche **25**) :

```
> cvm.test(serie,pexp,lambda)
```
[2]

Ajustement à une loi de χ^2 (voir fiche **26**) :

```
> cvm.test(serie,pchisq,ddl)
```
[2]

66. Comparaison de deux distributions

— PACKAGE(S) ADDITIONNEL(S) UTILISÉ(S) DANS CETTE FICHE —

[1]`cramer`, [2]`RVAideMemoire`

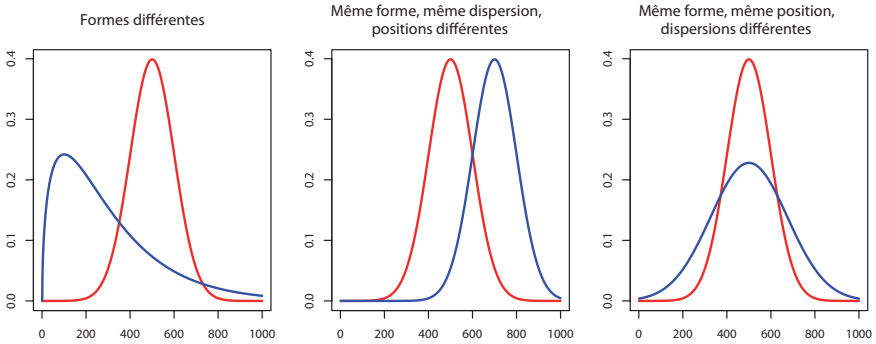
Lorsque l'on compare deux distributions, on peut s'intéresser à trois différences :

— une différence de *forme*.

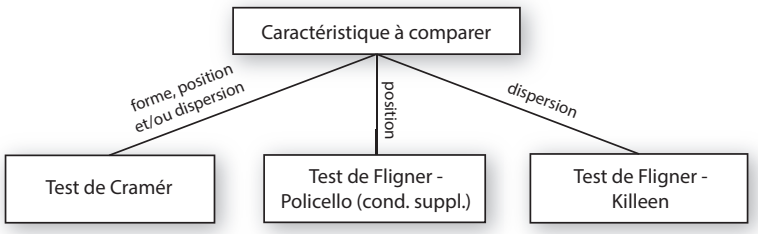
— une différence de *position*. Cette position peut parfois être résumée en un paramètre, comme la médiane (voir fiche **35**).

— une différence de *dispersion*. Cette dispersion peut parfois être résumée en un paramètre, comme la variance (voir fiche **36**).

Remarque : les tests sensibles à des différences de forme sont également sensibles à des différences de position et/ou de dispersion. Ils sont appelés *omnibus*, car ils testent les trois aspects simultanément.



Pour choisir le test approprié :



Test de Cramér (non paramétrique)

Pour réaliser le test : `cramer.test(serie1,serie2)`[1], où `serie1` et `serie2` sont des vecteurs contenant les deux séries de données (qui n'ont pas à être de même longueur).

Test de Fligner - Policello (non paramétrique)

Condition supplémentaire : la distribution des données doit être symétrique dans les deux groupes (mais peu importe si la forme est différente).

Pour réaliser le test : `fp.test(serie1,serie2)`[2]. Les deux séries de données ne doivent pas forcément être de même longueur.

Ce test n'est sensible ni à la forme, ni à la dispersion.

Test de Fligner - Killeen (non paramétrique)

Pour réaliser le test : `fligner.test(serie1,serie2)`. Les deux séries de données ne doivent pas forcément être de même longueur.

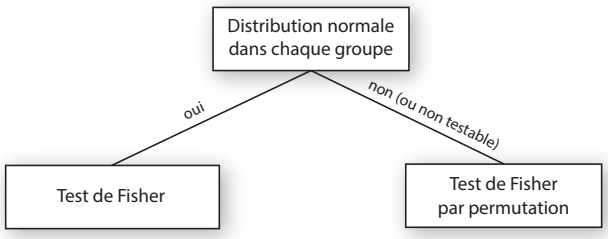
Ce test n'est pas sensible à la position. Il fonctionne d'autant mieux que les données sont distribuées normalement, mais est très robuste si ce n'est pas le cas.

67. Comparaison de deux variances

— PACKAGE(S) ADDITIONNEL(S) UTILISÉ(S) DANS CETTE FICHE —

[1]`RVAideMemoire`

Pour choisir le test approprié :



Remarque : le fait que plusieurs variances ne soient pas différentes est appelé « homogénéité des variances » ou « hétéroscédasticité ».

Test de Fisher (paramétrique)

Pour réaliser le test : `var.test(reponse~facteur)` où `reponse` est un vecteur contenant les valeurs de la variable réponse et `facteur` un vecteur contenant la modalité de chaque individu (dans le même ordre que `reponse`). Le symbole `~` signifie « expliqué par » ou « en fonction de ».

Test de Fisher par permutation (non paramétrique)

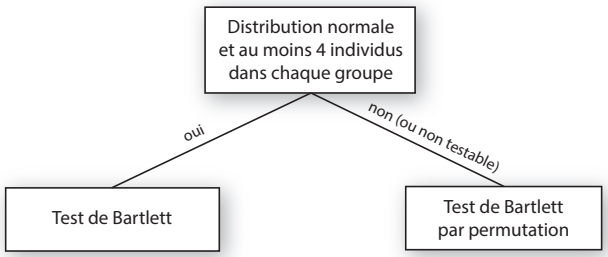
Pour réaliser le test : `perm.var.test(reponse~facteur)`[1].

68. Comparaison de plus de deux variances

— PACKAGE(S) ADDITIONNEL(S) UTILISÉ(S) DANS CETTE FICHE —

[1]`RVAideMemoire`

Pour choisir le test approprié :



Remarque : le fait que plusieurs variances ne soient pas différentes est appelé « homogénéité des variances » ou « hétéroscédasticité ».

Test de Bartlett (paramétrique)

Pour réaliser le test : `bartlett.test(reponse~facteur)` où `reponse` est un vecteur contenant les valeurs de la variable réponse et `facteur` un vecteur contenant la modalité de chaque individu (dans le même ordre que `reponse`). Le symbole `~` signifie « expliqué par » ou « en fonction de ».

Une *p-value* significative indique qu'au moins deux classes du facteur ont un effet différent sur la variable à expliquer (sans préciser lesquelles). Il est dans ce cas nécessaire de réaliser des comparaisons deux-à-deux pour identifier les classes en question, *via* `pairwise.var.test(reponse,facteur)`[1].

Il peut arriver que les comparaisons deux-à-deux n'indiquent aucune différence significative, contrairement au test global. Dans ce cas, la solution la plus prudente est de considérer qu'on ne peut pas savoir quelles variances sont responsables du rejet de l'hypothèse nulle dans le test global.

Test de Bartlett par permutation (non paramétrique)

Pour réaliser le test : `perm.bartlett.test(reponse~facteur)`[1].

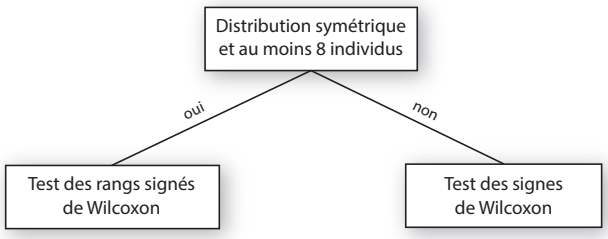
Si la *p-value* est significative, les comparaisons deux-à-deux sont réalisées *via* `pairwise.perm.var.test(reponse,facteur)`[1].

69. Conformité d'une médiane à une valeur théorique

— PACKAGE(S) ADDITIONNEL(S) UTILISÉ(S) DANS CETTE FICHE —

[1]`RVAideMemoire`

Pour choisir le test approprié :



Remarque : le caractère symétrique de la distribution se vérifie sur un histogramme (voir fiche **31**). Peu importe que la distribution soi uni- ou polymodale.

Test des rangs signés de Wilcoxon (non paramétrique)

Pour réaliser le test : `wilcox.test(serie,mu=m.theo)` où `serie` est un vecteur contenant la série de données et `m.theo` la médiane théorique.

Test des signes de Wilcoxon (non paramétrique)

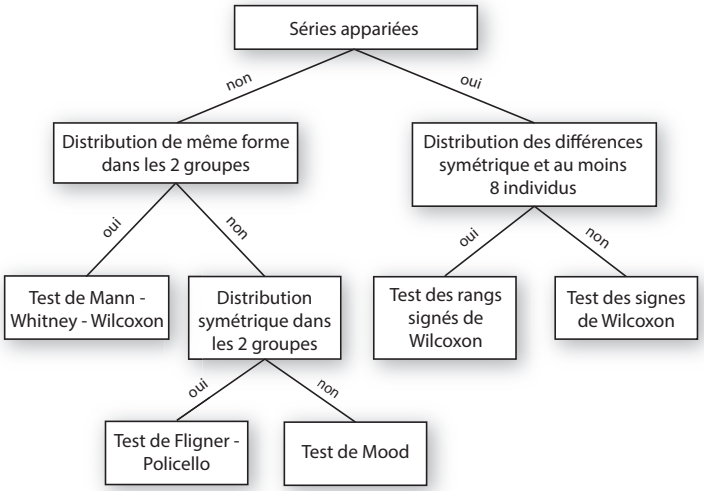
Pour réaliser le test : `wilcox.signtest(serie,mu=m.theo)`[1].

70. Comparaison de deux médianes

— PACKAGE(S) ADDITIONNEL(S) UTILISÉ(S) DANS CETTE FICHE —

[1]`RVAideMemoire`

Pour choisir le test approprié :



Remarques : la forme globale et le caractère symétrique d'une distribution se vérifient sur un histogramme (voir fiche **31**). Pour le test de Mann - Whitney - Wilcoxon les deux distributions doivent vraiment avoir quasiment la même forme (y compris la même dispersion, voir fiche **66** pour une illustration). Pour le test de Fligner - Policello peu importe la forme des deux distributions, pourvu qu'elles soient symétriques dans chacun des deux groupes. Pour les tests avec séries appariées, seule compte la forme de la distribution des *différences entre les valeurs appariées*. Ces différences sont obtenues *via* `differences<-reponse[as.numeric(facteur)==1]-reponse[as.numeric(facteur)==2]`, où `reponse` est un vecteur contenant les valeurs de la variable réponse et `facteur` un vecteur contenant la modalité de chaque individu (dans le même ordre que `reponse` et les valeurs appriées étant dans le même ordre dans les deux modalités).

Séries non appariées

Test de Mann - Whitney - Wilcoxon (non paramétrique)

Pour réaliser le test : `wilcox.test(reponse~facteur)`. Le symbole `~` signifie « expliqué par » ou « en fonction de ».

Test de Fligner - Policello (non paramétrique)

Pour réaliser le test : `fp.test(reponse~facteur)`[1].

Test de Mood (non paramétrique)

Pour réaliser le test : `mood.medtest(reponse~facteur)`[1].

Séries appariées

Test des rangs signés Wilcoxon (non paramétrique)

Pour réaliser le test : `wilcox.test(reponse~facteur,paired=TRUE)`.

Test des signes Wilcoxon (non paramétrique)

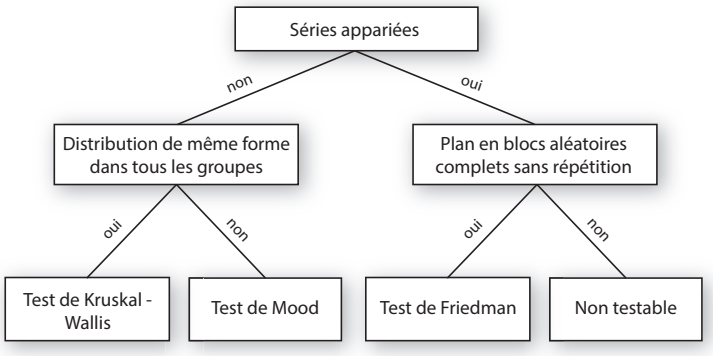
Pour réaliser le test : `wilcox.signtest(reponse~facteur)`[1].

71. Comparaison de plus de deux médianes

— PACKAGE(S) ADDITIONNEL(S) UTILISÉ(S) DANS CETTE FICHE —

[1]`FSA`, [2]`RVAideMemoire`

Pour choisir le test approprié :



Remarques : la forme globale et le caractère symétrique d'une distribution se vérifient sur un histogramme (voir fiche **31**). Pour le test de Kruskal - Wallis toutes les distributions doivent vraiment avoir quasiment la même forme (y compris la même dispersion, voir fiche **66** pour une illustration). Voir fiche **13** pour une explication de ce qu'est un plan d'expérience en blocs aléatoires complets sans répétition.

Séries non appariées

Test de Kruskal - Wallis (non paramétrique)

Pour réaliser le test : `kruskal.test(reponse~facteur)` où `reponse` est un vecteur contenant les valeurs de la variable réponse et `facteur` un vecteur contenant la modalité de chaque individu (dans le même ordre que `reponse`). Le symbole `~` signifie « expliqué par » ou « en fonction de ».

Une *p-value* significative indique qu'au moins deux classes du facteur ont un effet différent sur la variable à expliquer (sans préciser lesquelles). Il est dans ce cas nécessaire de réaliser des comparaisons deux-à-deux pour identifier les classes en question, *via* `dunnTest(reponse,facteur)`[1] (les *p-values* à prendre en compte sont celles de la colonne `P.adj`).

Il peut arriver que les comparaisons deux-à-deux n'indiquent aucune différence significative, contrairement au test global. Dans ce cas, la solution la plus prudente est de considérer qu'on ne peut pas savoir quelles médianes sont responsables du rejet de l'hypothèse nulle dans le test global.

Test de Mood (non paramétrique)

Pour réaliser le test : `mood.medtest(reponse~facteur)`[2].

Si la *p-value* est significative, les comparaisons deux-à-deux sont réalisées *via* `pairwise.mood.medtest(reponse,facteur)`[2].

Séries appariées

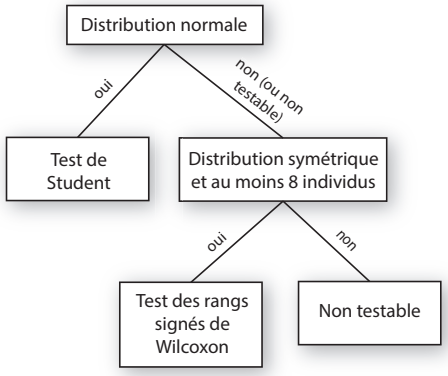
Test de Friedman (non paramétrique)

Pour réaliser le test : `friedman.test(reponse~fact.fixe|fact.alea)` où `fact.fixe` et `fact.alea` sont des vecteurs contenant la valeur de chaque individu (dans le même ordre) pour le facteur (fixe) dont on veut comparer les modalités et le facteur (aléatoire) servant à définir les séries appariées, respectivement.

Si la *p-value* est significative, les comparaisons deux-à-deux sont réalisées *via* `wilcox.paired.multcomp(variable~fact.fixe|fact.alea)`[2].

72. Conformité d'une moyenne à une valeur théorique

Pour choisir le test approprié :



Remarques : la normalité d'une distribution se vérifie grâce à des méthodes dédiées (voir fiche **65**) tandis que le caractère symétrique se vérifie sur un simple histogramme (voir fiche **31**). Pour le test des rangs signés de Wilcoxon, peu importe que la distribution soi uni- ou polymodale.

Test de Student (paramétrique)

Pour réaliser le test : `t.test(serie,mu=m.theo)` où `serie` est un vecteur contenant la série de données et `m.theo` la moyenne théorique.

Test des rangs signés de Wilcoxon (non paramétrique)

Pour réaliser le test : `wilcox.test(serie,mu=m.theo)`.

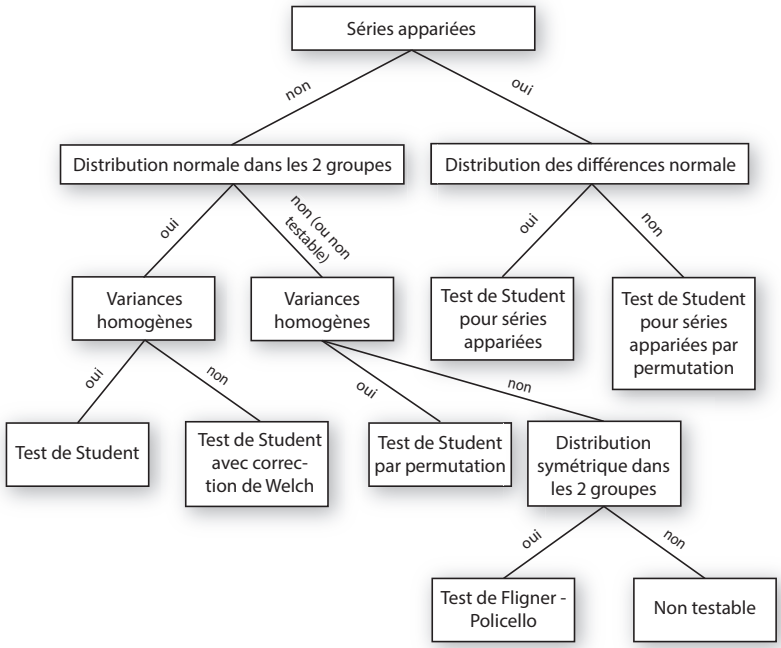
Remarque : ce test ne travaille en fait pas directement sur la moyenne. Mais si la distribution des données est symétrique, la conclusion du test peut s'appliquer à la moyenne.

73. Comparaison de deux moyennes

— PACKAGE(S) ADDITIONNEL(S) UTILISÉ(S) DANS CETTE FICHE —

`RVAideMemoire`[1]

Pour choisir le test approprié :



Remarques : la normalité d'une distribution se vérifie grâce à des méthodes dédiées (voir fiche **65**) tandis que le caractère symétrique se vérifie sur un simple histogramme (voir fiche **31**). Pour le test de Fligner - Policello peu importe la forme des deux distributions, pourvu qu'elles soient symétriques dans chacun des deux groupes. Pour les tests avec séries appariées, seule compte la forme de la distribution des *différences entre les valeurs appariées*. Ces différences sont obtenues *via* `differences<-reponse[as.numeric(facteur)==1]-reponse[as.numeric(facteur)==2]`, où `reponse` est un vecteur contenant les valeurs de la variable réponse et `facteur` un vecteur contenant la modalité de chaque individu (dans le même ordre que `reponse` et les valeurs appriées étant dans le même ordre dans les deux modalités).

Séries non appariées

Test de Student (paramétrique)

Pour réaliser le test : `t.test(reponse~facteur,var.equal=TRUE)`. Le symbole `~` signifie « expliqué par » ou « en fonction de ».

Test de Student avec correction de Welch (paramétrique)

Pour réaliser le test : `t.test(reponse~facteur,var.equal=FALSE)` ou plus simplement `t.test(variable~facteur)` (la correction est en fait appliquée par défaut).

Test de Student par permutation (non paramétrique)

Pour réaliser le test : `perm.t.test(reponse~facteur)`[1].

Test de Fligner - Policello (non paramétrique)

Pour réaliser le test : `fp.test(reponse~facteur)`[1].

Remarque : ce test compare en fait la *médiane* des deux échantillons. Mais si la distribution des données est symétrique dans chacun des deux groupes, médiane et moyenne sont très proches et la conclusion du test peut donc s'appliquer à la moyenne.

Séries appariées

Test de Student pour séries appariées (paramétrique)

Pour réaliser le test : `t.test(reponse~facteur,paired=TRUE)`.

Test de Student pour séries appariées par permutation (non paramétrique)

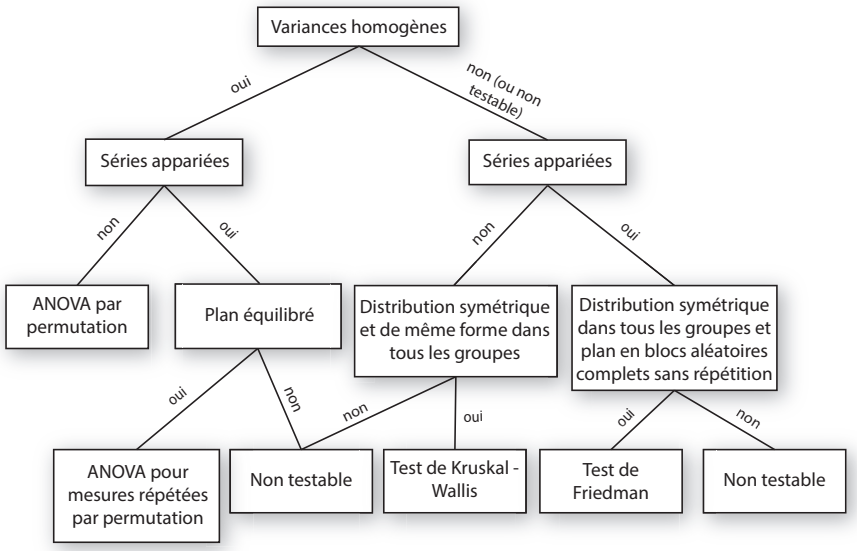
Pour réaliser le test : `perm.t.test(reponse~facteur,paired=TRUE)`[1].

74. Comparaison de plus de deux moyennes – 1 facteur

— PACKAGE(S) ADDITIONNEL(S) UTILISÉ(S) DANS CETTE FICHE —

[1]`RVAideMemoire`

Pour choisir le test approprié :



Remarques : la forme globale et le caractère symétrique d'une distribution se vérifient sur un histogramme (voir fiche **31**). Pour le test de Kruskal - Wallis toutes les distributions doivent vraiment avoir quasiment la même forme (y compris la même dispersion, voir fiche **66** pour une illustration). Pour l'ANOVA pour mesures répétées par permutation, « plan équilibré » signifie autant d'individus dans chaque combinaison du facteur fixe et du facteur alétoire. Voir fiche **13** pour une explication de ce qu'est un plan d'expérience en blocs aléatoires complets sans répétition.

Séries non appariées

ANOVA par permutation (non paramétrique)

Pour réaliser le test : `perm.anova(reponse~facteur)`[1], où `reponse` et `facteur` sont des vecteurs contenant la valeur de chaque individu pour chaque variable (dans le même ordre). Le symbole `~` signifie « expliqué par » ou « en fonction de ».

Une *p-value* significative indique qu'au moins deux classes du facteur ont un effet différent sur la variable à expliquer (sans préciser lesquelles). Il est dans ce cas nécessaire de réaliser des comparaisons deux-à-deux pour identifier les classes en question, *via* `pairwise.perm.t.test(reponse,facteur)`[1].

Il peut arriver que les comparaisons deux-à-deux n'indiquent aucune différence significative, contrairement au test global. Dans ce cas, la solution la plus prudente est de considérer qu'on ne peut pas savoir quelles moyennes sont responsables du rejet de l'hypothèse nulle dans le test global.

Test de Kruskal - Wallis (non paramétrique)

Pour réaliser le test : `kruskal.test(reponse~facteur)`.

Si la *p-value* est significative, les comparaisons deux-à-deux sont réalisées *via* `dunn.test(reponse, facteur)`[1].

Remarque : le test de Kruskal-Wallis et le test de Dunn pour les comparaisons multiples travaillent en fait sur la *médiane* des échantillons. Mais si la distribution des données est symétrique dans tous les groupes, médiane et moyenne sont très proches et la conclusion des tests peut donc s'appliquer à la

moyenne.

Séries appariées

ANOVA pour mesures répétées par permutation (non paramétrique)

Pour réaliser le test : `perm.anova(reponse~fact.fixe|fact.alea)`[1] où `fact.fixe` et `fact.alea` sont des vecteurs contenant la valeur de chaque individu (dans le même ordre) pour le facteur (fixe) dont on veut comparer les modalités et le facteur (aléatoire) servant à définir les séries appariées, respectivement.

Si la *p-value* est significative, les comparaisons deux-à-deux sont réalisées *via* `pairwise.perm.t.test(reponse,fact.fixe,paired=TRUE)`[1]. Attention les valeurs appariées doivent être dans le même ordre dans toutes les modalités de `fact.fixe`.

Test de Friedman (non paramétrique)

Pour réaliser le test : `friedman.test(reponse~fact.fixe|fact.alea)`.

Si la *p-value* est significative, les comparaisons deux-à-deux sont réalisées *via* `wilcox.paired.multcomp(reponse~fact.fixe|fact.alea)`[1].

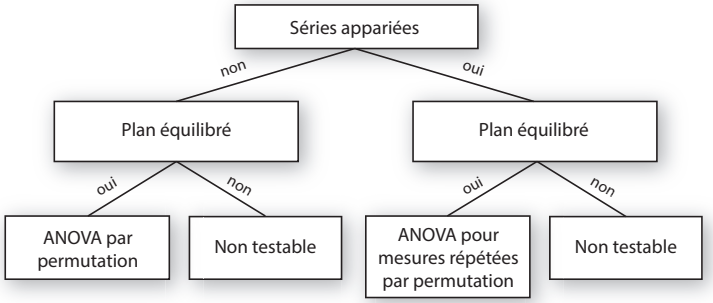
Remarque : le test de Friedman et le test des rangs signés de Wilcoxon pour les comparaisons multiples travaillent en fait sur la *médiane* des échantillons. Mais si la distribution des données est symétrique dans tous les groupes, médiane et moyenne sont très proches et la conclusion des tests peut donc s'appliquer à la moyenne.

75. Comparaison de plus de deux moyennes – 2 facteurs

— PACKAGE(S) ADDITIONNEL(S) UTILISÉ(S) DANS CETTE FICHE —

`[1]RVAideMemoire`

Pour choisir le test approprié :



Remarque : « plan équilibré » signifie autant d'individus dans chaque combinaison de tous les facteurs (deux pour l'ANOVA par permutation, trois (avec le facteur aléatoire) pour l'ANOVA pour mesures répétées par permutation).

Séries non appariées

ANOVA par permutation (non paramétrique)

Pour réaliser le test : `perm.anova(formule)`[1] où `formule` peut être (voir fiche **40** pour une explication détaillée de la construction d'une formule) :

- `reponse~fact1+fact2`
- `reponse~fact1*fact2`
- `reponse~fact1/fact2` pour un modèle où *les deux facteurs sont fixes*
- `reponse~fact1/fact2` en ajoutant l'argument `nest.f2="random"` pour un modèle où `fact2` *est un facteur aléatoire.*

Si une *p-value* est significative, cela indique qu'au moins deux classes du facteur en question (ou au moins deux combinaisons de classes des deux facteurs si c'est l'effet de l'interaction qui est significatif) ont un effet différent sur la variable à expliquer (sans préciser lesquelles). Il est dans ce cas nécessaire de réaliser des comparaisons deux-à-deux pour identifier les classes (ou combinaisons de classes) en question, *via* `pairwise.perm.t.test(reponse,facteur)`[1], où `facteur` est `fact1`, `fact2` ou `fact1:fact2` selon la *p-value* qui est significative.

Il peut arriver que les comparaisons deux-à-deux n'indiquent aucune différence significative, contrairement au test global. Dans ce cas, la solution la plus prudente est de considérer qu'on ne peut pas savoir quelles moyennes sont responsables du rejet de l'hypothèse nulle dans le test global.

Séries appariées

ANOVA pour mesures répétées par permutation (non paramétrique)

Pour réaliser le test : `perm.anova(formule)`[1] où `formule` peut être (voir fiche **40** pour une explication détaillée de la construction d'une formule) :

- `reponse~fact.fixe1+fact.fixe2|fact.alea`
- `reponse~fact.fixe1*fact.fixe2|fact.alea`.

Si une *p-value* est significative, les comparaisons deux-à-deux sont réalisées *via* `pairwise.perm.t.test(reponse~facteur,paired=TRUE)`, où `facteur` est `fact.fixe1`, `fact.fixe2` ou `fact.fixe1:fact.fixe2` selon la *p-value* qui est significative. Attention les valeurs appariées doivent être dans le même ordre dans toutes les modalités de `fact.fixe`.

76. Analyser une variable continue non bornée – relation(s) linéaire(s)

— PACKAGE(S) ADDITIONNEL(S) UTILISÉ(S) DANS CETTE FICHE —

[1]`lme4`, [2]`MASS`, [3]`MuMIn`, [4]`car`, [5]`RVAideMemoire`

⚠ L'analyse présentée dans cette fiche est basée sur un modèle. Il est indispensable de maîtriser les notions des fiches **39** à **42**. ⚠

Modèle utilisé

Si le principe de l'analyse est le même qu'il y ait des séries appariées ou non, quelques modalités diffèrent. La première est que sans séries appariées le modèle utilisé est un Modèle Linéaire (*Linear Model* ou LM), tandis qu'avec des séries appariées c'est un Modèle Linéaire Mixte (*Linear Mixed Model* ou LMM ; « mixte » sous-entendant « contenant au moins un facteur aléatoire », facteurs que l'on utilise précisément pour identifier les séries appariées (voir fiche **11**)).

Remarque : lorsque toutes les variables explicatives sont des covariables, on est dans le cas particulier d'une *régression linéaire* (au sens des moindres carrés).

Construction du modèle

Pour créer le modèle :

— sans séries appariées : `modele<-lm(formule)`

— avec des séries appariées : `modele<-lmer(formule)`[1].

Voir fiche **40** pour une explication détaillée de la construction d'une formule. De manière générale, on peut dire que :

— inclure un facteur permet de tester si les moyennes diffèrent entre les niveaux de ce facteur.

— inclure une covariable permet de tester s'il existe une relation entre cette covariable et la variable à expliquer.

— inclure une interaction entre deux variables explicatives permet de tester si l'effet de l'une dépend de l'autre (le raisonnement est le même pour une interaction incluant plus de deux termes). Le cas particulier d'une interaction entre une covariable et un facteur permet de tester si la relation entre la covariable et la variable à expliquer est différente selon la modalité du facteur.

Vérification de l'ajustement aux données

Avant d'aller plus loin, il est indispensable de vérifier que le modèle s'ajuste bien aux données. Cette étape est fondamentale, et ce pour tout modèle, car un test basé sur un modèle mal ajusté n'est simplement pas fiable (ni valide). Trois conditions sont à vérifier (dans cet ordre).

1. Indépendance entre les résidus du modèle et les valeurs prédites

Voir fiche **41** pour une explication détaillée et la manière de vérifier cette condition. Si elle n'est pas respectée de façon flagrante, plusieurs solutions sont possibles (à tester dans cet ordre) :

1. utiliser une *fonction de lien*, ce qui suppose de changer de modèle pour passer soit sur un Modèle Linéaire Généralisé (*Generalized Linear Model* ou GLM, `glm()` au lieu de `lm()`) s'il n'y a pas de séries appariées, soit sur un Modèle Linéaire Généralisé Mixte (*Generalized Linear Mixed Model* ou GLMM, `glmer()`[1] au lieu de `lmer()`[1]) s'il y a des séries appariées. Dans les deux cas le modèle est basé sur une loi normale, comme le LM(M) (ce qui est implicite dans `lm()` et `lmer()`[1] car un LM(M) est forcément basé sur une loi normale). Tester les fonctions de lien log (en ajoutant l'argument `family=gaussian(link="log")`) ou inverse ($\frac{1}{x}$, en ajoutant l'argument `family=gaussian(link="inverse")`), *i.e.* les deux autres acceptées par la loi de normale.
2. Si certaines variables *explicatives* sont quantitatives, rester sur le LM(M) et en transformer une ou plusieurs ($\sqrt{x}$ et $log(x)$ sont les transformations les plus fréquentes).

2. Équivariance des résidus

Voir fiche **41** pour une explication détaillée et la manière de vérifier cette condition.

Si l'hypothèse d'équivariance n'est pas respectée, plusieurs solutions sont possibles (à tester dans leur ordre respectif) :

— (G)LM : deux alternatives sont envisageables :

1. Modéliser la relation entre valeurs prédites et variance des résidus. En général, l'hypothèse d'équivariance n'est pas respectée car la variance des résidus augmente avec les valeurs prédites. Cela peut se modéliser grâce à un GLM basé sur une loi « quasi » (qui n'est en réalité pas vraiment une loi mais une sorte de « boîte à outils » très flexible), qui ne change pas la fonction de lien éventuellement définie à l'étape précédente :
   — Si le modèle était un LM, passer sur un GLM (`glm()` au lieu de `lm()`) et ajouter l'argument `family=quasi(link="identity",variance="mu")`.
   Remarque : la fonction de lien « identité » veut en fait dire qu'il n'y a pas de fonction de lien, ce qui est implicite dans `lm()` car un LM n'a forcément pas de fonction de lien.
   — Si le modèle était déjà un GLM, modifier l'argument `family` en `family=quasi(link=lien, variance="mu")` où `lien` est la fonction de lien définie à l'étape précédente.
   Si la variance des résidus augmente toujours avec les valeurs prédites, essayer de remplacer `variance="mu"` par `variance="mu^2"` (voire `variance="mu^3"` plus rarement).
2. Transformer la variable *à expliquer*. Trois options sont classiques : $\sqrt{x}$, $log(x)$ si la première transformation n'est pas suffisante, $\sqrt[4]{x}$ ($= x^{1/4}$) en alternative à la deuxième s'il y a des 0 dans les données. Attention, une transformation peut remettre en cause l'indépedance des résidus. Si une transformation est nécessaire pour stabiliser la variance des résidus, mieux vaut repartir du LM d'origine puis vérifier l'hypothèse d'indépendance avant de tester l'équivariance.

— (G)LMM : la démarche est identique à celle présentée pour un (G)LM. Elle passe par l'utilisation de la fonction `glmmPQL()`[2] : `modele<-glmmPQL(formule.fixe,random=formule.alea,family=quasi(link=lien,variance="mu"),verbose=FALSE)`[2].
Remarque : la particularité de la fonction `glmmPQL()`[2] est que les parties fixe et aléatoire de la formule sont séparées. La partie fixe (`formule.fixe`) se construit comme décrit dans la fiche **40**. La partie aléatoire est sous la forme `~1|fact.alea` où `fact.alea` est le facteur aléatoire (pour intégrer plusieurs facteurs aléatoires, utiliser `random=list(formule1,formule2...)` où chaque formule correspond à un facteur aléatoire).

3. Normalité des résidus

Voir fiche **41** pour une explication détaillée et la manière de vérifier cette condition. Elle n'est pas à respecter si les résidus ont une variance non homogène et que celle-ci a été modélisée correctement (voir condition précédente).

Si l'hypothèse de normalité n'est pas respectée de façon flagrante, transformer la variable *à expliquer* ($\sqrt{x}$, $log(x)$ ou $\sqrt[4]{x}$). Attention, une transformation peut remettre en cause l'indépedance et l'équivariance des résidus. Si une transformation est nécessaire pour stabiliser la variance des résidus, mieux vaut repartir du LM d'origine puis vérifier les hypothèses d'indépendance et d'équivariance avant de tester la normalité.

Remarque : cette condition est la moins importante des trois car les modèles sont robustes à un non-respect limité.

Capacité explicative globale

On peut estimer la capacité explicative globale d'un modèle grâce au *coefficient de détermination* (R^2), qui représente la proportion de la variance de la variable à expliquer qui est expliquée par les variables explicatives. Ce coefficient est obtenu :

— LM : *via* `summary(modele)$r.squared`

— LMM et GLMM : *via* `r.squaredGLMM(modele)`[3]. La fonction renvoie en fait deux valeurs : le R^2 *marginal* (`R2m`) qui correspond à la part de la variance expliquée uniquement par les facteurs fixes et covariables, et le R^2 *conditionnel* (`R2c`) qui correspond à la part de la variance expliquée par l'ensemble des variables explicatives (fixes et aléatoires).

Remarque : selon les cas, le coefficient de détermination n'est pas toujours disponible avec un GLM.

Test(s)

Selon le modèle, le même test n'est pas à utiliser :

— LM : le test est un test F réalisé *via* `Anova(modele)`[4] (voir fiche **42** pour une explication détaillée des hypothèses testées). Dans le cas où toutes les variables explicatives sont des facteurs ce test est appelé ANOVA (*ANalysis Of VAriance*).
Remarque : bien que le terme « ANOVA » soit utilisé abondamment, une ANOVA correspond en fait à une seule situation bien particulière : un test F sur un modèle linéaire au sens des moindres carrés ordinaires dont toutes les variables explicatives sont des facteurs.

— GLM : le test est un test F réalisé *via* `Anova(modele,test="F")`[4] (voir fiche **42** pour une explication détaillée des hypothèses testées).

— LMM ou GLMM : le test est un test de Wald – en fait un test par terme du modèle (*i.e.* un par ligne du tableau renvoyé) – réalisé *via* `Anova(modele)`[4] (voir fiche **42** pour une explication détaillée des hypothèses testées).

Si un facteur (ou une interaction impliquant un facteur) a un effet significatif, il est nécessaire de réaliser des comparaisons multiples pour identifier les modalités (ou combinaisons de modalités) qui diffèrent. Voir fiche **43** pour réaliser ces comparaisons.

Si une covariable a un effet significatif, la direction de son effet est donnée par le *signe* du paramètre associé (qui correspond à la *pente* de la droite reliant la covariable et la variable réponse, ininterprétable directement dans le cas d'un GLM(M)). Les valeurs de tous les paramètres du modèle sont obtenues *via* `summary(modele)`. Elles sont appelées `Estimate` et se trouvent dans le tableau `Coefficients` pour un LM ou un GLM, appelées `Estimate` et dans le tableau `Fixed effects` pour un LMM, appelées `Value` et dans le tableau `Fixed effects` pour un GLMM. Si le coefficient portant le nom de la covariable est négatif, la réponse diminue quand la valeur de la covariable augmente ; s'il est positif, la réponse augmente quand la valeur de la covariable augmente. L'erreur standard de tous les coefficients est donnée dans la colonne `Std.Error`.

Remarque : dans le cas d'un régression linéaire simple (*i.e.* LM avec seulement une covariable comme variable explicative), la valeur de l'ordonnée à l'origine du modèle peut être intéressante. Elle est donnée sur la ligne `(Intercept)`.

Prédiction à partir du modèle

L'intérêt des modèles est à la fois de comprendre l'importance des différentes variables explicatives sur la variable à expliquer, mais également de prédire la valeur que prendrait cette variable à expliquer pour des valeurs *connues* des variables explicatives. Prédire une réponse nécessite donc de fixer la valeur de *toutes* les variables explicatives.

Deux méthodes peuvent être utilisées pour la prédiction (seule la seconde est disponible pour les (G)LMMs), les deux étant basées sur la fonction `predict()` :

— donner la valeur (ou les valeurs pour faire plusieurs prédictions) de chacune des variables explicatives directement dans la fonction, sous la forme d'une liste : `predict(modele,newdata=list(variables))`, où `variables` est un enchaînement de `variable1=valeur,variable2=valeur...` Dans le cas d'un GLM ajouter l'argument `type="response"`.

— créer un tableau contenant une colonne par variable explicative (les noms de colonnes doivent être *strictement identiques* aux noms des variables du modèle), et remplir chaque ligne en fonction des valeurs pour lesquelles la prédiction doit être faite (il y a donc une prédiction par ligne de ce tableau). Puis : `predict(modele,newdata=tableau)`, avec l'argument `type="response"` pour un GLM(M).
Remarque : pour les (G)LMMs, les valeurs du (ou des) facteur(s) aléatoire(s) peuvent être données ou non. Si ce n'est pas le cas, il est nécessaire d'ajouter l'argument `re.form=NA` (LMM) ou `level=0` (GLMM) à la fonction `predict()`. Les prédictions prennent alors en compte l'effet moyen de toutes les modalités du (ou des) facteur(s) aléatoire(s) du modèle.

EXEMPLE(S)

Avec un modèle contenant un facteur à deux niveaux (A et B), une covariable variant de 0 à 30, et leur interaction :

```
> modele <- lm(reponse~facteur*covariable)
```

On peut prédire une moyenne de cette façon :
```
> predict(modele,newdata=list(facteur="A",covariable=10))
```
Ou, pour plusieurs prédictions :
```
> predict(modele,newdata=list(facteur=c("A","B"),covariable=c(10,10)))
```
Ou encore :
```
> predict(modele,newdata=list(facteur=c("A","B"),covariable=rep(10,2)))
```
Ou encore créer un tableau de ce type :
```
> tableau
  facteur covariable
1       A         10
2       B         10
```
Puis :
```
> predict(modele,newdata=tableau)
```

Graphes

Effet d'un facteur

Pour illustrer l'effet d'un facteur, on réalise généralement un diagramme en barres où sont représentées les moyennes par modalité. Deux types de moyennes peuvent être représentées :

— les moyennes brutes (*i.e.* calculées à partir des données brutes), avec leurs erreurs standards : voir fiches **35** et **37**, et l'utilisation de la fonction `tapply()`. Attention, dans le cas des LMMs, ces moyennes et erreurs standards ne tiennent pas compte du (des) facteur(s) aléatoire(s).

— les moyennes ajustées en fonction des autres variables du modèle, également avec leurs erreurs standards : voir fiche **43**.

Une fois les moyennes et erreurs standards récupérées, le diagramme peut être tracé (voir fiche **34**).

Relation avec une covariable

Illustrer la relation entre la variable à expliquer et une covariable nécessite trois étapes :

1. Tracer les points observés : `plot(reponse~covariable)`.
2. Créer un vecteur ayant les mêmes minimum et maximum que la covariable mais découpé en très petits intervalles : `x <- seq2(covariable)`[5].
3. Ajouter la droite de la relation sur le graphe, ajustée par le modèle. La détermination de tous les points de la droite est en fait basée sur une prédiction : la valeur que prend la variable à expliquer pour chaque valeur du vecteur `x`. C'est le grand nombre de valeurs de ce vecteur, et le très petit intervalle qui les sépare, qui donne l'aspect lissé de la droite (qui est en fait constituée de centaines de points reliés entre eux). Si le modèle contient d'autres variables explicatives que la covariable du graphe, il faut en fixer toutes les valeurs à l'identique (de sorte que seule la covariable du graphe change de valeur pour toutes les prédictions). La droite s'ajoute *via* `lines(x,predict(modele,newdata=variables.à.expliquer))` (avec l'argument `type="response"` à la fonction `predict()` dans le cas d'un GLM(M)).
   Remarque : la fonction `lines()` peut être utilisée plusieurs fois de suite pour tracer plusieurs droites sur le même graphe, par exemple pour plusieurs niveaux d'un même facteur contenu dans le modèle. Pour modifier le type de tracé et la couleur des droites, utiliser les arguments `lty` et `col` respectivement (pour plus d'informations et (beaucoup) plus d'options graphiques, voir `?par`). Enfin, pour ajouter une légende, utiliser la fonction `legend()` (voir `?legend` pour plus d'informations).

EXEMPLE(S)

On se base toujours sur le modèle suivant :
```
> modele <- lm(reponse~facteur*covariable)
```
Étape 1 : tracer les points correspondant aux données observées :
```
> plot(reponse~covariable)
```
Étape 2 : créer le vecteur `x` :
```
> x <- seq2(covariable)[5]
```

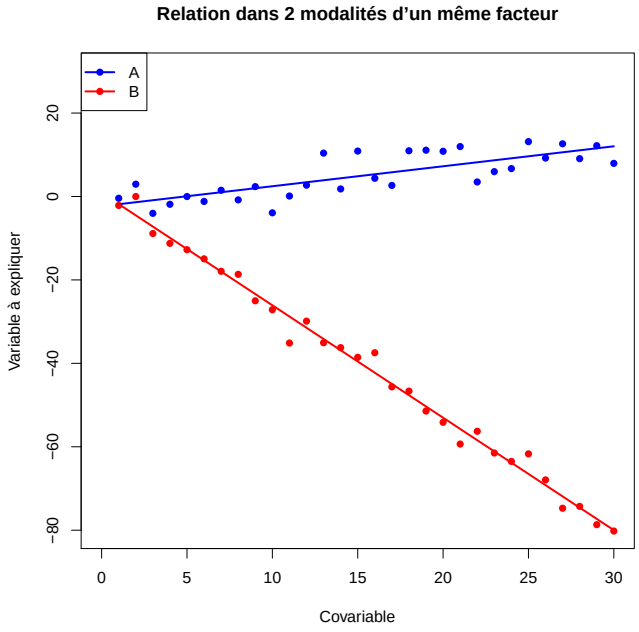
Étape 3 : ajouter la droite. On choisit de se placer dans la modalité A du facteur :

```
> lines(x,predict(modele,newdata=list(covariable=x,facteur=rep("A",length(x)))))
```

La fonction `rep()` est très utile dans cette situation, puisqu'elle permet de créer très simplement un vecteur de la même longueur que `x`, contenant une seule valeur répétée.

Pour ajouter la droite de la relation dans la modalité B :

```
> lines(x,predict(modele,newdata=list(covariable=x,facteur=rep("B",length(x)))))
```



77. Analyser une variable continue non bornée – relation non linéaire

— PACKAGE(S) ADDITIONNEL(S) UTILISÉ(S) DANS CETTE FICHE —

[1]`nlme`, [2]`RVAideMemoire`

Modèle utilisé

Le modèle utilisé est un Modèle Non Linéaire (*Non Linear Model* ou NLM). Il s'utilise le plus souvent avec des variables explicatives qui sont toutes des covariables ; ce cas particulier est appelé *régression non linéaire*. Un facteur peut être ajouté pour définir des *paramètres* différents par modalité.

Construction du modèle

L'utilisation des NLM exige deux choses :

— connaître l'équation qui lie la variable à expliquer et les covariables

— avoir une idée *a priori* de la valeur des paramètres de cette équation (la majeure partie du temps, on peut se faire cette idée à partir d'un graphe).

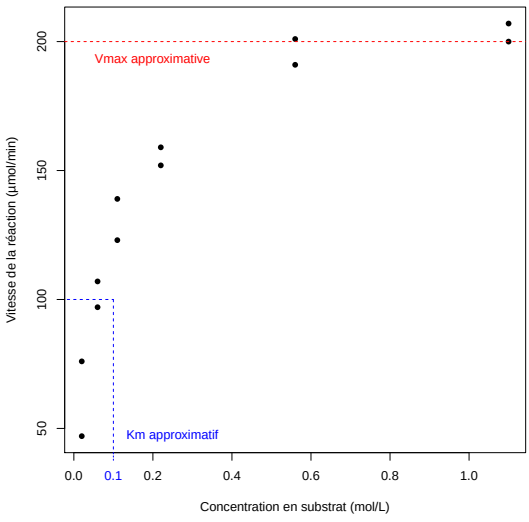
Le modèle peut ensuite être créé : `modele<-nls(reponse~equation,start=list(parametres))`, où `equation` est l'équation explicite de la relation et `parametres` la liste des paramètres de l'équation avec pour chacun une valeur approximative. Le symbole `~` signifie « expliqué par » ou « en fonction de ».

— EXEMPLE(S) —

L'équation de Michaelis - Menten sur la vitesse d'une réaction enzymatique est de la forme :

$$vitesse = \frac{V_{max} \cdot concentration}{K_M + concentration}$$

On a obtenu les points expérimentaux suivants :



D'après le graphe, on estime V_{max} à 200 μmol/min et K_M à 0,1 mol/L. Le modèle s'écrit donc :

```
> modele <- nls(vitesse~Vmax*concentration/(Km+concentration),start=list(Vmax=
200,Km=0.1))
```

Certaines équations communes ont été implémentées dans **R**, dans des fonctions qui permettent à la fois de ne pas préciser l'équation mais aussi (et surtout) d'estimer automatiquement la valeur *a priori* des paramètres :

Type de régression	**Équation**	`equation`
Asymptotique		
Michaelis - Menten	$y = \frac{Vm \cdot x}{K+x}$	`SSmicmen(x,Vm,K)`
Exponentielle à 2 paramètres	$y = Asym(1 - e^{-e^{lrc}x})$	`SSasympOrig(x,Asym,lrc)`
Exponentielle à 3 paramètres	$y = Asym + (R_0 - Asym)e^{-e^{lrc}x}$	`SSasymp(x,Asym,R0,lrc)`
Sigmoïde		
Logistique à 3 paramètres	$y = \frac{Asym}{1+e^{\frac{xmid-x}{scal}}}$	`SSlogis(x,Asym,xmid,scal)`
Logistique à 4 paramètres	$y = A + \frac{B-A}{1+e^{\frac{xmid-x}{scal}}}$	`SSfpl(x,A,B,xmid,scal)`
Weibull	$y = Asym - Drop \cdot e^{-e^{lrc}x^{pwr}}$	`SSweibull(x,Asym,Drop,lrc,pwr)`
Gompertz	$y = Asym \cdot e^{-b2 \cdot b3^x}$	`SSgompertz(x,Asym,b2,b3)`
En cloche		
Biexponentielle	$y = A1 \cdot e^{-e^{lrc1}x} + A2 \cdot e^{-e^{lrc2}x}$	`SSbiexp(x,A1,lrc1,A2,lrc2)`

Utiliser ces fonctions simplifie considérablement la construction du modèle.

EXEMPLE(S)

Sur le même modèle de Michaelis - Menten, notre premier modèle :

```
> modele <- nls(vitesse~Vmax*concentration/(Km+concentration),start=list(Vmax=200,Km=0.1))
```

est équivalent à :

```
> modele <- nls(vitesse~SSmicmen(concentration,Vmax,Km))
```

Pour estimer des paramètres différents selon les modalités d'un facteur, utiliser la fonction `nlsList()`[1] à la place de `nls()` (les deux fonctions sont basées sur la même syntaxe) et ajouter `|facteur` après `equation`, où `facteur` est un vecteur contenant la modalité de chaque individu (dans le même ordre que la variable à expliquer et les covariables).

EXEMPLE(S)

Toujours sur le même modèle de Michaelis - Menten, si l'on a réalisé l'expérience avec deux enzymes et que l'on veut estimer des paramètres différents pour chacune d'entre elles :

```
> modele <- nlsList(vitesse~Vmax*concentration/(Km+concentration)|enzyme,start=list(Vmax=200,Km=0.1))
```
[1]

ou :

```
> modele <- nlsList(vitesse~SSmicmen(concentration,Vmax,Km)|enzyme)
```
[1]

Vérification de l'ajustement aux données

Avant d'aller plus loin, il est indispensable de vérifier que le modèle s'ajuste bien aux données. Cette étape est fondamentale, et ce pour tout modèle, car un test basé sur un modèle mal ajusté n'est simplement pas fiable (ni valide). Voir fiche **41** pour une explication détaillée de cette vérification.

Si le modèle est de façon flagrante non valide, ce peut être car l'équation sur lequel il est basé est mal choisie auquel cas il faut la modifier. La variable à expliquer et/ou une (ou plusieurs) variable(s) explicative(s) peuvent également être transformées. Trois options sont fréquentes : $\sqrt{x}$, $log(x)$ si la première transformation n'est pas suffisante, $\sqrt[4]{x}$ $(= x^{1/4})$ en alternative à la deuxième s'il y a des 0 dans les données.

Récupération des paramètres et tests

Les valeurs de tous les paramètres du modèle sont obtenues *via* `summary(modele)`. Elles se trouvent dans le tableau `Parameters` pour un modèle sans facteur (créé avec `nls()`), `Coefficients` pour un

modèle avec un facteur (créé avec `nlsList()`[1]). Dans les deux cas, le tableau renvoie la valeur (`Estimate`) et l'erreur standard (`Std. Error`) de chaque paramètre, et teste leur conformité à la valeur nulle (une *p-value* non significative indique donc que le paramètre peut être supprimé de l'équation du modèle).

Prédiction à partir du modèle

L'intérêt des modèles est d'estimer les paramètres de la relation qui lie la variable à expliquer et les covariables, mais également de prédire la valeur que prendrait cette variable à expliquer pour des valeurs *connues* des variables explicatives. Prédire une réponse nécessite donc de fixer la valeur de *toutes* les variables explicatives.

Deux méthodes peuvent être utilisées pour la prédiction, les deux étant basées sur la fonction `predict()` :

— donner la valeur (ou les valeurs pour faire plusieurs prédictions) de chacune des variables explicatives directement dans la fonction, sous la forme d'une liste : `predict(modele,newdata=list(variables))`, où `variables` est un enchaînement de `variable1=valeur,variable2=valeur...`

— créer un tableau contenant une colonne par variable explicative (les noms de colonnes doivent être *strictement identiques* aux noms des variables du modèle), et remplir chaque ligne en fonction des valeurs pour lesquelles la prédiction doit être faite (il y a donc une prédiction par ligne de ce tableau). Puis : `predict(modele,newdata=tableau)`.

EXEMPLE(S)

Avec notre même modèle de Michaelis - Menten et nos deux enzymes (A et B) :

```
> modele <- nlsList(vitesse~SSmicmen(concentration,Vmax,Km)|enzyme)[1]
```

On peut prédire une valeur de vitesse de cette façon :

```
> predict(modele,newdata=list(enzyme="A",concentration=0.5))
```

Ou, pour plusieurs prédictions :

```
> predict(modele,newdata=list(enzyme=c("A","B"),concentration=c(0.5,0.5)))
```

Ou encore :

```
> predict(modele,newdata=list(enzyme=c("A","B"),concentration=rep(0.5,2)))
```

Ou encore créer un tableau de ce type :

```
> tableau
  enzyme concentration
1      A           0.5
2      B           0.5
```

Puis :

```
> predict(modele,newdata=tableau)
```

Graphes

Illustrer la relation entre la variable à expliquer et une covariable nécessite trois étapes :

1. Tracer les points observés : `plot(reponse~covariable)`.
2. Créer un vecteur ayant les mêmes minimum et maximum que la covariable mais découpé en très petits intervalles : `x <- seq2(covariable)`[2].
3. Ajouter la courbe de la relation sur le graphe, ajustée par le modèle. La détermination de tous les points de la courbe est en fait basée sur une prédiction : la valeur que prend la variable à expliquer pour chaque valeur du vecteur `x`. C'est le grand nombre de valeurs de ce vecteur, et le très petit intervalle qui les sépare, qui donne l'aspect lissé de la courbe (qui est en fait constituée de centaines de points reliés entre eux). Si le modèle contient d'autres variables explicatives que la covariable du graphe, il faut en fixer toutes les valeurs à l'identique (de sorte que seule la covariable du graphe change de valeur pour toutes les prédictions). La courbe s'ajoute *via* `lines(x,predict(modele,newdata=variables.à.expliquer))`.
   Remarque : la fonction `lines()` peut être utilisée plusieurs fois de suite pour tracer plusieurs courbes sur le même graphe, par exemple pour plusieurs niveaux d'un même facteur contenu dans le modèle. Pour modifier le type de tracé et la couleur des courbes, utiliser les arguments `lty` et `col` respectivement (pour plus d'informations et (beaucoup) plus d'options graphiques,

voir `?par`). Enfin, pour ajouter une légende, utiliser la fonction `legend()` (voir `?legend` pour plus d'informations).

EXEMPLE(S)

On se base toujours sur le modèle suivant :

```
> modele <- nlsList(vitesse~SSmicmen(concentration,Vmax,Km)|enzyme)
```
[1]

Étape 1 : tracer les points correspondant aux données observées :

```
> plot(vitesse~concentration)
```

Étape 2 : créer le vecteur `x` :

```
> x <- seq2(concentration)
```
[2]

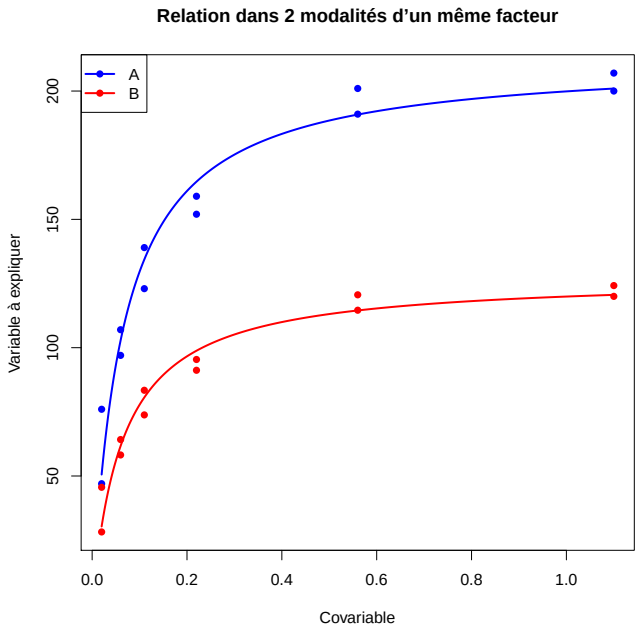
Étape 3 : ajouter la courbe. On choisit de se placer dans la modalité A du facteur :

```
> lines(x,predict(modele,newdata=list(concentration=x,enzyme=rep("A",length(x))
)))
```

La fonction `rep()` est très utile dans cette situation, puisqu'elle permet de créer très simplement un vecteur de la même longueur que `x`, contenant une seule valeur répétée.

Pour ajouter la courbe de la relation dans la modalité B :

```
> lines(x,predict(modele,newdata=list(concentration=x,enzyme=rep("B",length(x))
)))
```



78. Choisir le modèle d'analyse d'un temps de survie

— PACKAGE(S) ADDITIONNEL(S) UTILISÉ(S) DANS CETTE FICHE —

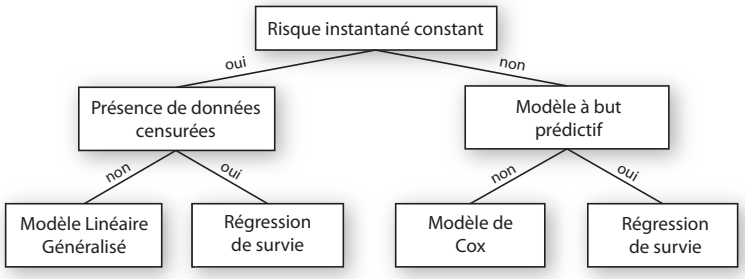
[1]`RVAideMemoire`

Avant toute analyse de temps de survie, il est indispensable d'avoir bien compris les notions suivantes :

— *Censure* : un individu est dit censuré lorsque la date de référence pour calculer le temps avant la mort (*i.e.* le « point de départ ») n'est pas connue (censure *à gauche*), ou que la mort n'a pas été observée avant la fin de l'étude (parce que l'étude s'est arrêtée ou parce que l'individu en est sorti ; censure *à droite*). Seules les censures à droite sont traitées dans cet aide-mémoire. Une condition essentielle pour la prise en compte des données censurées dans l'analyse est qu'elles doivent être *indépendantes* des conditions d'expérience.

— *Risque instantané* : ce risque est celui de mourir à l'instant t, sachant que la mort n'est pas survenue avant. Il peut être constant, ou au contraire augmenter ou diminuer avec le temps.

Plusieurs modèles assez différents peuvent être utilisés pour analyser des temps de survie. Pour choisir le plus approprié :



Remarque : la régression de survie peut parfaitement être utilisée pour interpréter des données sans qu'il y ait pour objectif de faire de la prédiction. Le modèle de Cox, lui par contre, est restreint à l'interprétation d'un jeu de données particulier.

Pour savoir si le risque instantané est constant, tracer la courbe de survie des individus *via* `plotsurvivors(mort,censure)`[1] où `mort` est un vecteur contenant le délai avant la mort de chaque individu et `censure` un vecteur indiquant si l'individu est censuré ou non (0 si censuré ou 1 si non censuré, *i.e.* 0 si la mort n'a pas été observée ou 1 si elle l'a été), dans le même ordre que `mort`. Le risque instantané est considéré comme constant si les points de la courbe de survie sont à peu près alignés sur une droite.

79. Analyser un temps de survie – Modèle Linéaire Généralisé

— PACKAGE(S) ADDITIONNEL(S) UTILISÉ(S) DANS CETTE FICHE —

[1]`lme4`, [2]`car`, [3]`survival`, [4]`RVAideMemoire`

⚠ L'analyse présentée dans cette fiche est basée sur un modèle. Il est indispensable de maîtriser les notions des fiches **39** à **42**. ⚠

Réponse

La réponse (appelée « temps de survie dans cette fiche ») ne peut être que nulle ou positive.

Modèle utilisé

Si le principe de l'analyse est le même qu'il y ait des séries appariées ou non, quelques modalités diffèrent. La première est que sans séries appariées le modèle utilisé est un Modèle Linéaire Généralisé (*Generalized Linear Model* ou GLM), tandis qu'avec des séries appariées c'est un Modèle Linéaire Généralisé Mixte (*Generalized Linear Mixed Model* ou GLMM ; « mixte » sous-entendant « contenant au moins un facteur aléatoire », facteurs que l'on utilise précisément pour identifier les séries appariées (voir fiche **11**)).

Contrairement au Modèle Linéaire (et sa variante Mixte, voire fiche **76**), les GLM(M)s ne sont pas basés sur une loi normale. Dans le cas de temps de survie, la loi à utiliser est une loi Gamma.

Construction du modèle

Pour créer le modèle :

— sans séries appariées : `modele<-glm(formule,family="Gamma")`
— avec des séries appariées : `modele<-glmer(formule,family="Gamma")`[1].

Voir fiche **40** pour une explication détaillée de la construction d'une formule. De manière générale, on peut dire que :

— inclure un facteur permet de tester si les temps de survie diffèrent entre les niveaux de ce facteur.
— inclure une covariable permet de tester s'il existe une relation entre cette covariable et la variable à expliquer.
— inclure une interaction entre deux variables explicatives permet de tester si l'effet de l'une dépend de l'autre (le raisonnement est le même pour une interaction incluant plus de deux termes). Le cas particulier d'une interaction entre une covariable et un facteur permet de tester si la relation entre la covariable et la variable à expliquer est différente selon la modalité du facteur.

Vérification de l'ajustement aux données

Avant d'aller plus loin, il est indispensable de vérifier que le modèle s'ajuste bien aux données. Cette étape est fondamentale, et ce pour tout modèle, car un test basé sur un modèle mal ajusté n'est simplement pas fiable (ni valide).

La seule condition importante à vérifier est l'indépendance des résidus du modèle avec les valeurs prédites (voir fiche **41** pour une explication détaillée). Si cette hypothèse n'est pas respectée de façon flagrante, changer la *fonction de lien* sur laquelle est basé le modèle. Par défaut cette fonction de lien est l'inverse (*i.e.* $\frac{1}{x}$) et n'est pas précisée (`family="Gamma"` est un raccourci pour `family=Gamma(link="inverse")`). Tester les fonctions de lien log (`family=Gamma(link="log")`), ou identité (`family=Gamma(link="identity")`), *i.e.* les deux autres acceptées par la loi Gamma.

Test(s)

Quel que soit le modèle, l'effet des variables explicatives est testé par la même fonction : `Anova(modele)`[2] (voir fiche **42** pour une explication détaillée des hypothèses testées). Cependant, ce ne sont pas les mêmes tests qui sont réalisés selon le modèle :

— GLM : la fonction réalise un test du rapport des vraisemblances (*Likelihood Ratio Test* ou LR Test) – en fait un test par terme du modèle (*i.e.* un par ligne du tableau renvoyé).
— GLMM : la fonction réalise un test de Wald – en fait un test par terme du modèle (*i.e.* un par ligne du tableau renvoyé).

Si un facteur (ou une interaction impliquant un facteur) a un effet significatif, il est nécessaire de réaliser des comparaisons multiples pour identifier les modalités (ou combinaisons de modalités) qui diffèrent. Voir fiche **43** pour réaliser ces comparaisons.

Si une covariable a un effet significatif, la direction de son effet est donnée par le *signe* du paramètre associé. Les valeurs de tous les paramètres du modèle sont obtenues *via* `summary(modele)`. Elles sont appelées `Estimate` et se trouvent dans le tableau `Coefficients` pour un GLM, `Fixed effects` pour un GLMM. Si le coefficient portant le nom de la covariable est négatif, le temps de survie diminue quand la valeur de la covariable augmente ; s'il est positif, le temps de survie augmente quand la valeur de la covariable augmente.

Prédiction à partir du modèle

L'intérêt des modèles est à la fois de comprendre l'importance des différentes variables explicatives sur la variable à expliquer, mais également de prédire la valeur que prendrait cette variable à expliquer pour des valeurs *connues* des variables explicatives. Prédire un temps de survie nécessite donc de fixer la valeur de *toutes* les variables explicatives.

Deux méthodes peuvent être utilisées pour la prédiction (seule la seconde est disponible pour les GLMMs), les deux étant basées sur la fonction `predict()` :

— donner la valeur (ou les valeurs pour faire plusieurs prédictions) de chacune des variables explicatives directement dans la fonction, sous la forme d'une liste : `predict(modele,newdata=list(variables),type="response")`, où `variables` est un enchaînement de `variable1=valeur,variable2=valeur...`
— créer un tableau contenant une colonne par variable explicative (les noms de colonnes doivent être *strictement identiques* aux noms des variables du modèle), et remplir chaque ligne en fonction des valeurs pour lesquelles la prédiction doit être faite (il y a donc une prédiction par ligne de ce tableau). Puis : `predict(modele,newdata=tableau,type="response")`.
Remarque : pour les GLMMs, les valeurs du (ou des) facteur(s) aléatoire(s) peuvent être données ou non. Si ce n'est pas le cas, il est nécessaire d'ajouter l'argument `re.form=NA` à la fonction `predict()`. Les prédictions prennent alors en compte l'effet moyen de toutes les modalités du (ou des) facteur(s) aléatoire(s) du modèle.

EXEMPLE(S)

Avec un modèle contenant un facteur à deux niveaux (A et B), une covariable variant de 0 à 30, et leur interaction :

```
> modele <- glm(reponse~facteur*covariable,family="Gamma")
```

On peut prédire un temps de survie de cette façon :

```
> predict(modele,newdata=list(facteur="A",covariable=10),type="response")
```

Ou, pour plusieurs prédictions :

```
> predict(modele,newdata=list(facteur=c("A","B"),covariable=c(10,10)),type="response")
```

Ou encore :

```
> predict(modele,newdata=list(facteur=c("A","B"),covariable=rep(10,2)),type="response")
```

Ou encore créer un tableau de ce type :

```
> tableau
  facteur covariable
1       A         10
2       B         10
```

Puis :

```
> predict(modele,newdata=tableau,type="response")
```

Graphes

Effet d'un facteur

Pour illustrer l'effet d'un facteur, on réalise généralement des courbes de survie (dites de Kaplan - Meier). Leur tracé nécessite trois étapes :

1. Créer un *objet de survie*, une forme particulière des temps de survie : `survie<-Surv(reponse)`[3] où `reponse` est la variable à expliquer du modèle, *i.e.* les temps de survie.
2. Créer les données du graphe : `courbes<-survfit(survie~facteur)`[3].
3. Tracer le graphe : `plot(courbes)`. Il est possible d'ajouter l'intervalle de confiance (à 95 %) de chaque courbe en ajoutant l'argument `conf.int=TRUE` (attention cependant à ne pas multiplier les courbes, qui rendent le graphe difficilement lisible). Pour modifier le type de tracé et la couleur des courbes, utiliser les arguments `lty` et `col` respectivement (pour plus d'informations et (beaucoup) plus d'options graphiques, voir `?par`). Enfin, pour ajouter une légende, utiliser la fonction `legend()` (voir `?legend` pour plus d'informations).

Relation avec une covariable

Illustrer la relation entre la variable à expliquer et une covariable nécessite trois étapes :

1. Tracer les points observés : `plot(reponse~covariable)`.
2. Créer un vecteur ayant les mêmes minimum et maximum que la covariable mais découpé en très petits intervalles : `x <- seq2(covariable)`[4].
3. Ajouter la courbe de la relation sur le graphe, ajustée par le modèle. La détermination de tous les points de la courbe est en fait basée sur une prédiction : la valeur que prend la variable à expliquer pour chaque valeur du vecteur `x`. C'est le grand nombre de valeurs de ce vecteur, et le très petit intervalle qui les sépare, qui donne l'aspect lissé de la courbe (qui est en fait constituée de centaines de points reliés entre eux). Si le modèle contient d'autres variables explicatives que la covariable du graphe, il faut en fixer toutes les valeurs à l'identique (de sorte que seule la covariable du graphe change de valeur pour toutes les prédictions). La courbe s'ajoute *via* `lines(x,predict(modele,newdata=variables.à.expliquer,type="response"))`.
   Remarque : la fonction `lines()` peut être utilisée plusieurs fois de suite pour tracer plusieurs courbes sur le même graphe, par exemple pour plusieurs niveaux d'un même facteur contenu dans le modèle. Pour modifier le type de tracé et la couleur des courbes, utiliser les arguments `lty` et `col` respectivement (pour plus d'informations et (beaucoup) plus d'options graphiques, voir `?par`). Enfin, pour ajouter une légende, utiliser la fonction `legend()` (voir `?legend` pour plus d'informations).

EXEMPLE(S)

On se base toujours sur le modèle suivant :

```
> modele <- glm(reponse~facteur*covariable,family="Gamma")
```

Étape 1 : tracer les points correspondant aux données observées :

```
> plot(reponse~covariable)
```

Étape 2 : créer le vecteur `x` :

```
> x <- seq2(covariable)[4]
```

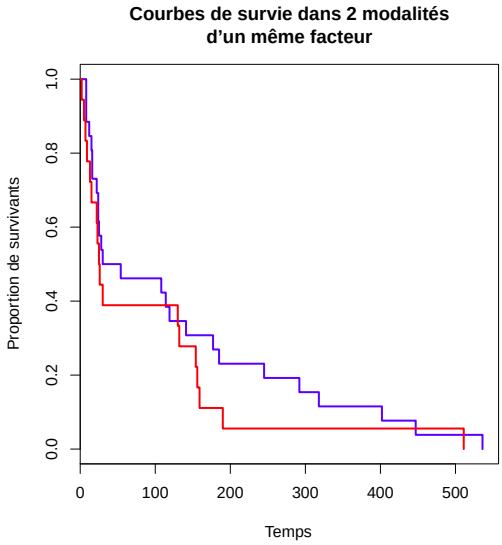
Étape 3 : ajouter la courbe. On choisit de se placer dans la modalité A du facteur :

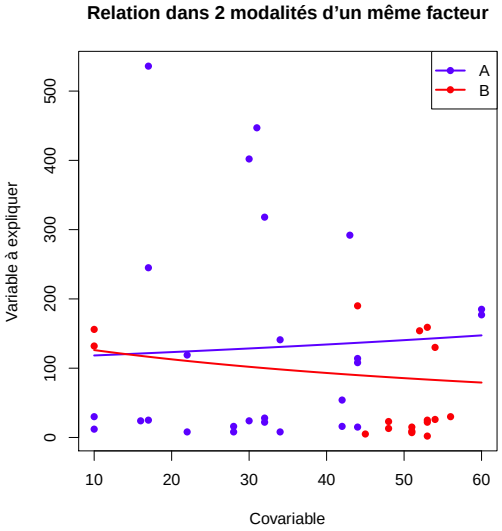
```
> lines(x,predict(modele,newdata=list(covariable=x,facteur=rep("A",length(x))),
type="response"))
```

La fonction `rep()` est très utile dans cette situation, puisqu'elle permet de créer très simplement un vecteur de la même longueur que `x`, contenant une seule valeur répétée.

Pour ajouter la courbe de la relation dans la modalité B :

```
> lines(x,predict(modele,newdata=list(covariable=x,facteur=rep("B",length(x))),
type="response"))
```


Courbes de survie dans 2 modalités
d'un même facteur
Proportion de survivants
0.0
0.2
0.4
0.6
0.8
1.0
0
100
200
300
400
500
Temps


Relation dans 2 modalités d'un même facteur
Variable à expliquer
0
100
200
300
400
500
10
20
30
40
50
60
Covariable
A
B

80. Analyser un temps de survie – Régression de survie

— PACKAGE(S) ADDITIONNEL(S) UTILISÉ(S) DANS CETTE FICHE —

[1]`survival`, [2]`car`, [3]`RVAideMemoire`

⚠ L'analyse présentée dans cette fiche est basée sur un modèle. Il est indispensable de maîtriser les notions des fiches **39** à **42**. ⚠

Réponse

Dans une régression de survie, la variable à expliquer n'est pas directement le temps avant la mort des individus, mais un objet particulier appelé *objet de survie*. Cet objet prend en compte à la fois le temps de survie, mais aussi des possibles censures (*i.e.* le fait que la mort ne soit pas observée avant la fin de l'étude ; voir fiche **78**). Pour le créer :

— `survie<-Surv(mort)`[1] s'il n'y a aucun individu censuré, où `mort` est le vecteur contenant les temps de survie.

— `survie<-Surv(mort,censure)`[1] s'il y a des individus censurés, où `mort` est le vecteur contenant les temps de survie et `censure` le vecteur contenant l'indication de censure de chaque individu (0 : censure, 1 : pas de censure *i.e.* mort observée).

Dans la formule du modèle, la variable à expliquer est `survie`, l'objet de survie.

Construction du modèle

Pour créer le modèle :

— si le risque instantané est constant (voir fiche **78** pour tester cette hypothèse) : `modele<-survreg(formule,dist="exponential")`[1]. La loi exponentielle sert à modéliser la constance du risque instantané.

— si le risque instantané n'est pas constant : `modele<-survreg(formule,dist="weibull")`[1]. La loi de Weibull sert à modéliser l'évolution du risque instantané au cours du temps (qu'il diminue ou qu'il augmente).

Voir fiche **40** pour une explication détaillée de la construction d'une formule. De manière générale, on peut dire que :

— inclure un facteur permet de tester si les temps de survie diffèrent entre les niveaux de ce facteur.

— inclure une covariable permet de tester s'il existe une relation entre cette covariable et la variable à expliquer.

— inclure une interaction entre deux variables explicatives permet de tester si l'effet de l'une dépend de l'autre (le raisonnement est le même pour une interaction incluant plus de deux termes). Le cas particulier d'une interaction entre une covariable et un facteur permet de tester si la relation entre la covariable et la variable à expliquer est différente selon la modalité du facteur.

Dans le cas d'un modèle avec risque instantané non constant, l'évolution de ce risque est donnée par `summary(modele)`. Si le paramètre `Scale` est < 1 le risque diminue avec le temps, s'il est > 1 il augmente.

Test(s)

Quel que soit le modèle, l'effet des variables explicatives est testé par la même fonction : `Anova(modele)`[2] (voir fiche **42** pour une explication détaillée des hypothèses testées). Le test réalisé est un test du rapport des vraisemblances (*Likelihood Ratio Test* ou LR Test) – en fait un test par terme du modèle (*i.e.* un par ligne du tableau renvoyé).

Si un facteur (ou une interaction impliquant un facteur) a un effet significatif, il est nécessaire de réaliser des comparaisons multiples pour identifier les modalités (ou combinaisons de modalités) qui diffèrent. Voir fiche **43** pour réaliser ces comparaisons.

Si une covariable a un effet significatif, la direction de son effet est donnée par le *signe* du paramètre associé. Les valeurs de tous les paramètres du modèle sont obtenues *via* `summary(modele)`. Elles sont appelées `Value`. Si le coefficient portant le nom de la covariable est négatif, le temps de survie diminue

quand la valeur de la covariable augmente ; s'il est positif, le temps de survie augmente quand la valeur de la covariable augmente.

Prédiction à partir du modèle

L'intérêt des modèles est à la fois de comprendre l'importance des différentes variables explicatives sur la variable à expliquer, mais également de prédire la valeur que prendrait cette variable à expliquer pour des valeurs *connues* des variables explicatives. Prédire un temps de survie nécessite donc de fixer la valeur de *toutes* les variables explicatives.

Deux méthodes peuvent être utilisées pour la prédiction, les deux étant basées sur la fonction `predict()` :

— donner la valeur (ou les valeurs pour faire plusieurs prédictions) de chacune des variables explicatives directement dans la fonction, sous la forme d'une liste : `predict(modele,newdata=list(variables))`, où `variables` est un enchaînement de `variable1=valeur,variable2=valeur...`

— créer un tableau contenant une colonne par variable explicative (les noms de colonnes doivent être *strictement identiques* aux noms des variables du modèle), et remplir chaque ligne en fonction des valeurs pour lesquelles la prédiction doit être faite (il y a donc une prédiction par ligne de ce tableau). Puis : `predict(modele,newdata=tableau)`.

EXEMPLE(S)

Avec un modèle contenant un facteur à deux niveaux (A et B), une covariable variant de 0 à 30, et leur interaction :

```
> modele <- survreg(survie~facteur*covariable,dist="exponential")
```
[1]

On peut prédire un temps de survie de cette façon :

```
> predict(modele,newdata=list(facteur="A",covariable=10))
```

Ou, pour plusieurs prédictions :

```
> predict(modele,newdata=list(facteur=c("A","B"),covariable=c(10,10)))
```

Ou encore :

```
> predict(modele,newdata=list(facteur=c("A","B"),covariable=rep(10,2)))
```

Ou encore créer un tableau de ce type :

```
> tableau
  facteur covariable
1       A         10
2       B         10
```

Puis :

```
> predict(modele,newdata=tableau)
```

Graphes

Effet d'un facteur

Pour illustrer l'effet d'un facteur, on réalise généralement des courbes de survie (dites de Kaplan - Meier). Leur tracé nécessite deux étapes :

1. Créer les données du graphe : `courbes<-survfit(survie~facteur)`[1].
2. Tracer le graphe : `plot(courbes)`. Si des individus sont censurés, ils sont représentés par des croix (+). Il est possible d'ajouter l'intervalle de confiance (à 95 %) de chaque courbe en ajoutant l'argument `conf.int=TRUE` (attention cependant à ne pas multiplier les courbes, qui rendent le graphe difficilement lisible). Pour modifier le type de tracé et la couleur des courbes, utiliser les arguments `lty` et `col` respectivement (pour plus d'informations et (beaucoup) plus d'options graphiques, voir `?par`). Enfin, pour ajouter une légende, utiliser la fonction `legend()` (voir `?legend` pour plus d'informations).

Relation avec une covariable

Illustrer la relation entre la variable à expliquer et une covariable nécessite trois étapes :

1. Tracer les points observés : `plot(mort~covariable)`.
2. Créer un vecteur ayant les mêmes minimum et maximum que la covariable mais découpé en très petits intervalles : `x <- seq2(covariable)`[3].

3. Ajouter la courbe de la relation sur le graphe, ajustée par le modèle. La détermination de tous les points de la courbe est en fait basée sur une prédiction : la valeur que prend la variable à expliquer pour chaque valeur du vecteur `x`. C'est le grand nombre de valeurs de ce vecteur, et le très petit intervalle qui les sépare, qui donne l'aspect lissé de la courbe (qui est en fait constituée de centaines de points reliés entre eux). Si le modèle contient d'autres variables explicatives que la covariable du graphe, il faut en fixer toutes les valeurs à l'identique (de sorte que seule la covariable du graphe change de valeur pour toutes les prédictions). La courbe s'ajoute *via* `lines(x,predict(modele,newdata=variables.à.expliquer))`.
   Remarque : la fonction `lines()` peut être utilisée plusieurs fois de suite pour tracer plusieurs courbes sur le même graphe, par exemple pour plusieurs niveaux d'un même facteur contenu dans le modèle. Pour modifier le type de tracé et la couleur des courbes, utiliser les arguments `lty` et `col` respectivement (pour plus d'informations et (beaucoup) plus d'options graphiques, voir `?par`). Enfin, pour ajouter une légende, utiliser la fonction `legend()` (voir `?legend` pour plus d'informations).

EXEMPLE(S)

On se base toujours sur le modèle suivant :

```
> modele <- survreg(survie~facteur*covariable,dist="exponential")
```
[1]

Étape 1 : tracer les points correspondant aux données observées :

```
> plot(mort~covariable)
```

Étape 2 : créer le vecteur `x` :

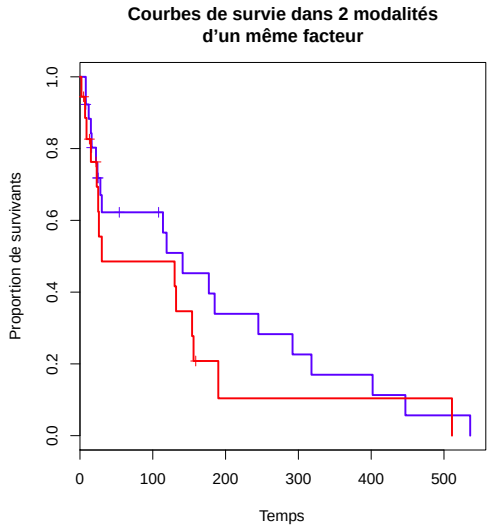
```
> x <- seq2(covariable)
```
[3]

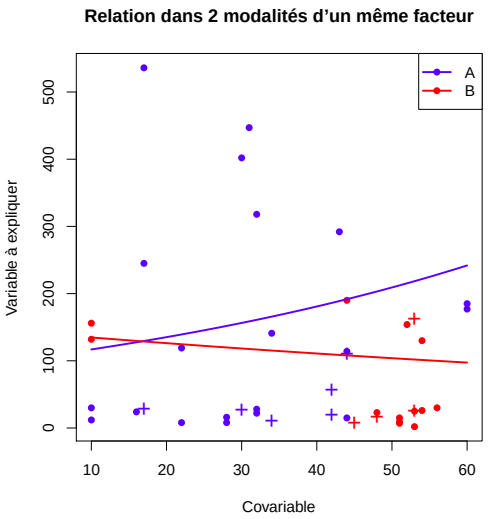
Étape 3 : ajouter la courbe. On choisit de se placer dans la modalité A du facteur :

```
> lines(x,predict(modele,newdata=list(covariable=x,facteur=rep("A",length(x)))
))
```

La fonction `rep()` est très utile dans cette situation, puisqu'elle permet de créer très simplement un vecteur de la même longueur que `x`, contenant une seule valeur répétée.
Pour ajouter la courbe de la relation dans la modalité B :

```
> lines(x,predict(modele,newdata=list(covariable=x,facteur=rep("B",length(x)))
))
```





81. Analyser un temps de survie – Modèle de Cox

— PACKAGE(S) ADDITIONNEL(S) UTILISÉ(S) DANS CETTE FICHE —

[1]`survival`, [2]`RVAideMemoire`, [3]`car`

⚠ L'analyse présentée dans cette fiche est basée sur un modèle. Il est indispensable de maîtriser les notions des fiches **39** à **42**. ⚠

Réponse

Dans un modèle de Cox, la variable à expliquer n'est pas directement le temps avant la mort des individus, mais un objet particulier appelé *objet de survie*. Cet objet prend en compte à la fois le temps de survie, mais aussi des possibles censures (*i.e.* le fait que la mort ne soit pas observée avant la fin de l'étude ; voir fiche **78**). Pour le créer :

- `survie<-Surv(mort)`[1] s'il n'y a aucun individu censuré, où `mort` est le vecteur contenant les temps de survie.
- `survie<-Surv(mort,censure)`[1] s'il y a des individus censurés, où `mort` est le vecteur contenant les temps de survie et `censure` le vecteur contenant l'indication de censure de chaque individu (0 : censure, 1 : pas de censure *i.e.* mort observée).

Dans la formule du modèle, la variable à expliquer est `survie`, l'objet de survie.

Construction du modèle

Pour créer le modèle : `modele<-coxph(formule)`[1]. Voir fiche **40** pour une explication détaillée de la construction d'une formule. De manière générale, on peut dire que :

- inclure un facteur permet de tester si les temps de survie diffèrent entre les niveaux de ce facteur.
- inclure une covariable permet de tester s'il existe une relation entre cette covariable et la variable à expliquer.
- inclure une interaction entre deux variables explicatives permet de tester si l'effet de l'une dépend de l'autre (le raisonnement est le même pour une interaction incluant plus de deux termes). Le cas particulier d'une interaction entre une covariable et un facteur permet de tester si la relation entre la covariable et la variable à expliquer est différente selon la modalité du facteur.

Le modèle de Cox est un peu particulier puisque, contrairement aux autres modèles d'analyse de temps de survie, il n'est pas paramétrique mais « semi » paramétrique, voire non paramétrique selon les points de vue (c'est la raison pour laquelle on ne peut pas l'utiliser dans un but prédictif).

Vérification de l'ajustement aux données

Avant d'aller plus loin, il est indispensable de vérifier que le modèle s'ajuste bien aux données. Cette étape est fondamentale, et ce pour tout modèle, car un test basé sur un modèle mal ajusté n'est simplement pas fiable (ni valide). Le modèle de Cox a des conditions de validité particulières :

- la relation entre chaque variable explicative quantitative (ou covariable) et le risque instantané doit être log-linéaire. Pour tester cette hypothèse : `cox.resid(modele)`[2]. La fonction trace un graphe par covariable, sur lequel la ligne rouge représente la tendance du nuage de point. On accepte l'hypothèse de log-linéarité pour une covariable si la ligne rouge correspondante est à peu près horizontale. Dans le cas contraire la solution la plus simple est de la découper en classes, puis la réintégrer dans le modèle comme facteur.

— EXEMPLE(S) —

Avec un modèle contenant un facteur et une covariable :

```
> modele <- coxph(survie~facteur+covariable)[1]
```

L'hypothèse de log-linéarité entre la variable à expliquer et la covariable est testée par :

```
> cox.resid(modele)[2]
```

Si l'hypothèse n'est pas respectée, on découpe la covariable en classes. Un exemple avec un découpage en deux classes :

```
> covar.class <- cut(covariable,breaks=2)
```

Remarque : pour ajouter un nom aux classes, utiliser l'argument `label`. Les classes ont par défaut la même longueur. Pour plus d'informations (beaucoup d'options sont disponibles), voir `?cut`.

Un nouveau modèle est créé avec la covariable transformée en facteur :

```
> modele <- coxph(survie~facteur+covar.class)[1]
```

— le rapport des risques instantanés de deux individus doit être constant au cours du temps (hypothèse des « risques proportionnels »). Pour tester cette hypothèse : `cox.zph(modele)`[1]. La fonction teste l'hypothèse pour chaque variable explicative, ainsi que pour le modèle global. Si une *p-value* est significative, cela indique que l'hypothèse n'est pas respectée pour la variable explicative en question, qui est dite *dépendante du temps*. Il vaut mieux alors l'intégrer dans le modèle en temps que *strate* et non variable explicative (pour les variables explicatives quantitatives, cela passe par un découpage en classes et une transformation en facteur). L'effet de la variable ne sera plus calculé, mais pris en compte à travers la définition de risques instantanés *de base* différents selon les strates. Pour intégrer une strate dans la formule du modèle, ajouter `+strata(variable)` après les variables explicatives (et retirer la variable désormais stratifiée des variables explicatives).

EXEMPLE(S)

Toujours avec le modèle suivant :

```
> modele <- coxph(survie~facteur+covariable)[1]
```

L'hypothèse des risques proportionnels est testée par :

```
> cox.zph(modele)[1]
```

Si l'hypothèse n'est pas respectée pour le facteur, le modèle est recréé en traitant celui-ci comme une strate :

```
> modele <- coxph(survie~covariable+strata(facteur))[1]
```

Si l'hypothèse n'est pas respectée pour la covariable, celle-ci doit être factorisée puis intégrée comme strate dans le modèle :

```
> modele <- coxph(survie~facteur+strata(covar.class))[1]
```

Test(s)

Selon que le modèle contienne une strate ou non, un test différent est à réaliser :

— modèle sans strate : `Anova(modele)`[3]. La fonction réalise un test du rapport des vraisemblances partielles (*Partial Likelihood Ratio Test* ou PLR Test) – en fait un test par terme du modèle (*i.e.* un par ligne du tableau renvoyé).

— modèle avec strate : `Anova(modele,test="Wald")`[3]. La fonction réalise un test de Wald – en fait un test par terme du modèle (*i.e.* un par ligne du tableau renvoyé).

Si un facteur (ou une interaction impliquant un facteur) a un effet significatif, il est nécessaire de réaliser des comparaisons multiples pour identifier les modalités (ou combinaisons de modalités) qui diffèrent. Voir fiche **43** pour réaliser ces comparaisons.

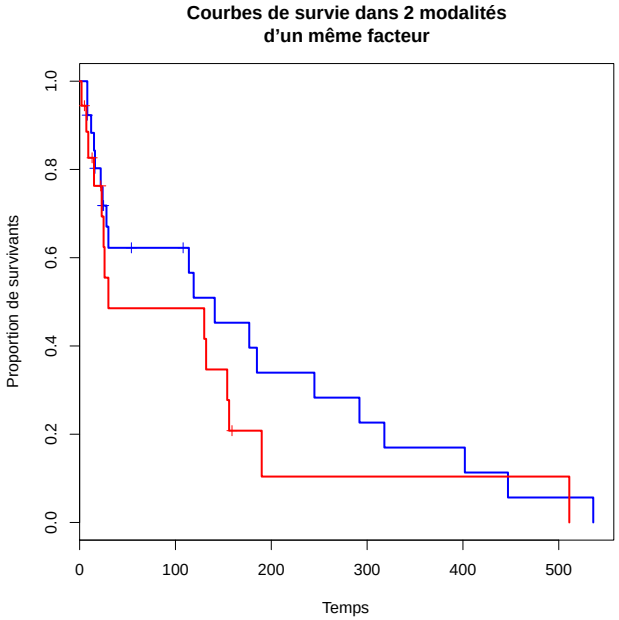
Si une covariable a un effet significatif, la direction de son effet est donnée par le *signe* du paramètre associé. Les valeurs de tous les paramètres du modèle sont obtenues *via* `summary(modele)`. Elles sont appelées `coef`. Si le coefficient portant le nom de la covariable est négatif, le temps de survie diminue quand la valeur de la covariable augmente ; s'il est positif, le temps de survie augmente quand la valeur de la covariable augmente.

Graphes

Le modèle de Cox n'étant pas prédictif, on illustre en pratique uniquement l'effet de facteurs sur le temps de survie. Pour ce faire, on réalise généralement des courbes de survie (dites de Kaplan - Meier). Leur tracé nécessite deux étapes :

1. Créer les données du graphe : `courbes<-survfit(survie~facteur)`[1]. Il est possible d'ajouter une strate dans la formule (sous la forme `+strata(facteur2)`) pour tracer une courbe par modalité du facteur et par strate. Attention cependant à ne pas multiplier les courbes, qui rendent le graphe difficilement lisible.

2. Tracer le graphe : `plot(courbes)`. Si des individus sont censurés, ils sont représentés par des croix (+). Il est possible d'ajouter l'intervalle de confiance (à 95 %) de chaque courbe en ajoutant l'argument `conf.int=TRUE` (même avertissement sur le nombre de courbes). Pour modifier le type de tracé et la couleur des courbes, utiliser les arguments `lty` et `col` respectivement (pour plus d'informations et (beaucoup) plus d'options graphiques, voir `?par`). Enfin, pour ajouter une légende, utiliser la fonction `legend()` (voir `?legend` pour plus d'informations).



82. Nuages de points

— PACKAGE(S) ADDITIONNEL(S) UTILISÉ(S) DANS CETTE FICHE —

[1]RVAideMemoire

Ce type de graphique permet de représenter les valeurs de deux variables numériques pour chaque individu, dans un graphe du type $y = f(x)$. Il permet d'avoir un aperçu de la relation qui peut exister entre ces variables. Il peut être utilisé avec deux vecteurs, une matrice à deux colonnes ou un tableau à deux colonnes. Son tracé est basé sur la fonction `plot()`.

Pour représenter deux vecteurs `x` et `y` contenant la valeur de chaque individu pour les deux variables (dans le même ordre) : `plot(y~x)`. Le symbole `~` signifie « expliqué par » ou « en fonction de ».

Pour ajouter un titre au graphe, utiliser l'argument `main="Titre"`.
Pour modifier la légende de l'axe horizontal, utiliser l'argument `xlab="Légende"`.
Pour modifier la légende de l'axe vertical, utiliser l'argument `ylab="Légende"`.
Pour (beaucoup) plus d'options graphiques, voir `?par`.

Pour ajouter une droite du type $y = ax + b$: `abline(b,a)`.

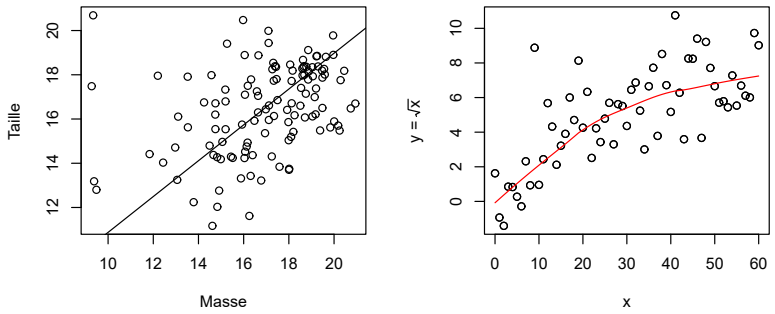
Pour ajouter une droite de régression linéaire au sens des moindres carrés (voir fiche **76**) : `abline(lm(y~x))`.

Pour ajouter une droite de régression linéaire au sens des moindres rectangles (voir fiche **87**) : `abline(least.rect(y~x)`[1]`)`.

Pour ajouter une courbe de tendance du nuage de points : `panel.smooth(x,y)`.

Pour ajouter une droite horizontale : `abline(h=ordonnee)`.

Pour ajouter une droite verticale : `abline(v=abscisse)`.



83. Intensité de la liaison entre deux variables

— PACKAGE(S) ADDITIONNEL(S) UTILISÉ(S) DANS CETTE FICHE —

[1]`RVAideMemoire`

Les paramètres suivants permettent de réduire deux séries de données à une valeur globale sur la liaison qui (éventuellement) les unit. Le choix du paramètre dépend de la nature des deux variables et parfois de la forme de leur relation. Celle-ci se vérifie grâce à un graphe de type nuage de points (voir fiche **82**) et peut être linéaire, monotone (*i.e.* uniquement croissante ou décroissante, quelle qu'en soit la manière) ou autre.

Deux variables quantitatives

Relation linéaire

Les deux paramètres les plus courants sont :

— la covariance de Pearson (paramétrique) : `cov(serie1,serie2)` où `serie1` et `serie2` sont des vecteurs contenant la valeur de chaque individu pour chaque variable (dans le même ordre)

— le coefficient de corrélation de Pearson (paramétrique) : `cor(serie1,serie2)`. Un cas particulier de la corrélation est la corrélation partielle (*i.e.* la corrélation entre deux variables tout en contrôlant pour une ou plusieurs autres variables). Son coefficient se calcule *via* `pcor(serie1,serie2,autres)`[1] où `autres` est un vecteur, un tableau ou une liste donnant les variables dont on veut retirer l'effet avant d'estimer la corrélation entre `serie1` et `serie2`. Dans tous les cas, un coefficient de corrélation varie entre -1 (corrélation parfaite et négative) et 1 (corrélation parfaite et positive), en passant par 0 (pas de corrélation).

Relation monotone

On retrouve les équivalents non paramétriques des paramètres précédents :

— covariance de Spearman (non paramétrique) : `cov(serie1,serie2,method="spearman")`

— coefficient de corrélation de Spearman (non paramétrique) : `cor(serie1,serie2,method="spearman")` ; la version partielle : `pcor(serie1,serie2,autres,method="spearman")`[1].

Relation autre

Aucun paramètre simple n'est disponible. Il est nécessaire de découper les variables en classes puis de les traiter comme qualitatives. Voir pour cela `?cut`.

Deux variables qualitatives ordinales

Relation monotone

On peut utiliser les versions non paramétriques (*i.e.* de Spearman) de la covariance et du coefficient de corrélation.

Relation autre

Aucun paramètre simple n'est disponible. Il est nécessaire de découper les variables en classes puis de les traiter comme qualitatives. Voir pour cela `?cut`.

Deux variables qualitatives

Si chaque classe des deux variables contient au moins 5 % du nombre total d'individus, on peut calculer la valeur du coefficient d'association de Cramér (non paramétrique) : `cramer(serie1,serie2)`[1]. Le coefficient varie :

— entre 0 (pas d'association ou *indépendance*) et 1 (association parfaite) dès que l'une des deux variables au moins a plus de deux classes.

— entre -1 et 1 quand les deux variables ont chacune deux classes. Peu importe dans quelle direction varie le coefficient, plus il s'éloigne de 0 et plus l'association est élevée.

84. Corrélation entre deux variables quantitatives ou ordinales

— PACKAGE(S) ADDITIONNEL(S) UTILISÉ(S) DANS CETTE FICHE —

[1]`RVAideMemoire`

L'objectif est ici de déterminer l'intervalle de confiance d'un coefficient de corrélation et de tester sa conformité à une valeur théorique. Voir fiche **83** pour une explication de quand utiliser le coefficient de Pearson (paramétrique) ou celui de Spearman (non paramétrique).

Pour toutes les fonctions calculant un intervalle de confiance, la précision de cet intervalle peut être modifiée grâce à l'argument `conf.level` (par défaut `conf.level=0.95`, ce qui calcule l'intervalle de confiance à 95 %).

Coefficient de corrélation de Pearson (paramétrique)

Intervalle de confiance et test de conformité à la valeur nulle

Pour réaliser le test : `cor.test(serie1,serie2)` où `serie1` et `serie2` sont des vecteurs contenant la valeur de chaque individu pour chaque variable (dans le même ordre). La fonction renvoie, dans l'ordre : le test de conformité à la valeur nulle (il y a « vraiment » corrélation si le résultat est significatif), l'intervalle de confiance et la valeur du coefficient.

Pour une corrélation partielle (*i.e.* où l'effet d'une ou plusieurs variables confondantes est retiré avant d'estimer et tester la corrélation) : `pcor.test(serie1,serie2,autres)`[1] où `autres` est un vecteur, un tableau ou une liste donnant les variables confondantes (dans le même ordre que `serie1` et `serie2`).

Test de conformité à une valeur autre que 0

Pour réaliser le test : `cor.conf(serie1,serie2,theo=valeur)`[1] où `valeur` est le coefficient de corrélation théorique.

Coefficient de corrélation de Spearman (non paramétrique)

Intervalle de confiance

L'intervalle de confiance est calculé par bootstrap : `spearman.ci(serie1,serie2)`[1].

Test de conformité à la valeur nulle

Pour réaliser le test : `cor.test(serie1,serie2,method="spearman")`. La fonction renvoie, dans l'ordre : le test de conformité à la valeur nulle et la valeur du coefficient.

Pour une corrélation partielle : `pcor.test(serie1,serie2,autres,method="spearman")`[1].

Test de conformité à une valeur autre que 0

Il n'existe pas de test dédié au coefficient de corrélation de Spearman. Cependant il suffit de calculer l'intervalle de confiance du coefficient et de voir si le coefficient théorique en fait partie. Par définition, si l'intervalle de confiance à 95 % ne comprend pas le coefficient théorique, alors le coefficient de corrélation observé est significativement différent du coefficient théorique au seul $\alpha = 5$ % (voir fiche **15**).

85. Comparaison de plusieurs coefficients de corrélation

— PACKAGE(S) ADDITIONNEL(S) UTILISÉ(S) DANS CETTE FICHE —

[1]RVAideMemoire

Voir fiche **83** pour une explication de quand utiliser le coefficient de Pearson (paramétrique) ou celui de Spearman (non paramétrique).

Coefficient de corrélation de Pearson (paramétrique)

Comparaison de deux coefficients

Pour réaliser le test, utiliser selon la situation l'une des deux fonctions suivantes :

— `cor.2comp(serie1,serie2,serie3,serie4)`[1] où `serie1` et `serie2` sont des vecteurs contenant la valeur de chaque individu pour les deux variables définissant la première corrélation (dans le même ordre), tandis que `serie3` et `serie4` sont des vecteurs contenant la valeur de chaque individu pour les deux variables définissant la seconde corrélation (dans le même ordre).

— `cor.multcomp(serie1,serie2,facteur)`[1] où `serie1` et `serie2` sont des vecteurs contenant la valeur de chaque individu pour les deux variables à tester (dans le même ordre), et `facteur` un facteur contenant la modalité de chaque individu.

Si les deux coefficients de corrélation ne sont pas significativement différents, les deux fonctions renvoient la valeur du coefficient de corrélation commun, son intervalle de confiance à 95 % et le résultat du test de conformité de ce coefficient à la valeur nulle (cette valeur théorique peut être modifiée grâce à l'argument `theo=valeur` où `valeur` est le coefficient de corrélation théorique).

Comparaison de plus de deux coefficients

Pour réaliser le test : `cor.multcomp(serie1,serie2,facteur)`[1].

Si la *p-value* du test est significative, cela indique qu'au moins deux coefficients diffèrent l'un de l'autre, sans préciser lesquels. La fonction effectue alors toutes les comparaisons deux-à-deux possibles.

Il peut arriver que les comparaisons deux-à-deux n'indiquent aucune différence significative, contrairement au test global. Dans ce cas, la solution la plus prudente est de considérer qu'on ne peut pas savoir quels coefficients sont responsables du rejet de l'hypothèse nulle dans le test global.

Coefficient de corrélation de Spearman (non paramétrique)

Il n'existe pas de test dédié au coefficient de corrélation de Spearman. Cependant il suffit de comparer les intervalles de confiance des différents coefficients. En effet, si deux intervalles de confiance 95 % ne se chevauchent pas alors les deux coefficients de corrélation sont significativement différents au seul $\alpha = 5$ % (voir fiche **15**). Comme on réalise des comparaisons multiples, les intervalles de confiance doivent être corrigés, comme on le ferait pour des *p-values* (voir fiche **16**).

Pour calculer les intervalles de confiance (corrigés) : `spearman.cor.multcomp(serie1,serie2,facteur)`[1]. La fonction renvoie un tableau à trois colonnes : les coefficients de corrélation (`r`) et les bornes inférieures (`inf`) et supérieures (`sup`) des intervalles de confiance.

— EXEMPLE(S) —

On obtient les trois intervalles de confiance suivants :

```
     inf      r    sup
A -0.978 -0.846 -0.419
B  0.241  0.789  0.957
C  0.593  0.904  0.985
```

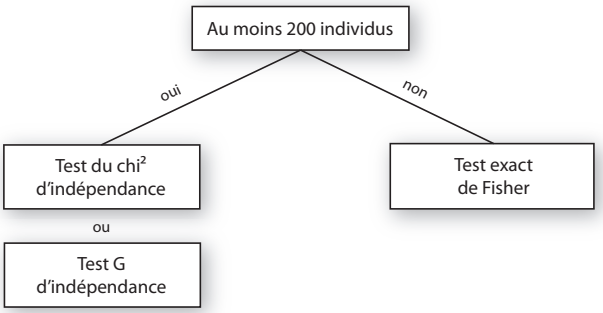
L'intervalle du groupe A ne chevauche pas ceux des groupes B et C, donc A est significativement différent de B et C. Les intervalles des groupes B et C, eux, se chevauchent. Donc B et C ne sont pas significativement différents.

86. Association entre deux variables qualitatives

— PACKAGE(S) ADDITIONNEL(S) UTILISÉ(S) DANS CETTE FICHE —

[1]`RVAideMemoire`

Pour choisir le test approprié :



Le test exact est toujours le plus fiable mais son temps de calcul augmente avec le nombre d'individus. Lorsque l'effectif est suffisamment grand, l'approximation faite par les tests du χ^2 et G est assez satisfaisante pour qu'ils puissent être utilisés. Les résultats de ces deux tests sont très semblables ; choisir entre l'un et l'autre relève plus d'une habitude que d'une raison statistique.

Les données doivent être organisées en un tableau de contingence du type :

		Variable B		
		Classe 1	…	Classe c
Variable A	Classe 1			
	…			
	Classe k			

où chaque case contient le nombre d'individus possédant à la fois le caractère de la variable A et celui de la variable B.

Ce tableau est obtenu de la manière suivante : `tab.cont<-table(variableA,variableB)` où `variableA` et `variableB` sont des vecteurs contenant la valeur de chaque individu pour chaque variable (dans le même ordre).

Test exact de Fisher (non paramétrique)

Pour réaliser le test : `fisher.test(tab.cont)`.

Si le message d'avertissement `out of workspace` apparaît, augmenter la valeur de l'argument `workspace` (par défaut `workspace=200000`). Si un autre message d'avertissement apparaît, cela peut être à cause d'un tableau trop complexe à analyser. Il faut alors se rabattre sur le test du χ^2 ou le test G.

Une *p-value* significative indique que les deux variables ne sont pas indépendantes, sans préciser les classes qui sont à l'origine de cette liaison. Il est dans ce cas nécessaire de réaliser des comparaisons deux-à-deux pour identifier les classes en question, *via* `fisher.multcomp(tab.cont)`[1]. La fonction réalise un test exact de Fisher sur chaque tableau de contingence 2 x 2 possible à partir de `tab.cont`. Il est nécessaire d'interpréter ces résultats pour repérer les classes qui apparaissent systématiquement dans les tests qui donnent une *p-value* significative. Ce sont ces classes qui sont liées.

Il peut arriver que les comparaisons deux-à-deux n'indiquent aucune liaison significative, contrairement au test global. Dans ce cas, la solution la plus prudente est de considérer qu'on ne peut pas savoir quelles classes sont responsables du rejet de l'hypothèse nulle dans le test global.

EXEMPLE(S)

Les caractères étudiés sont la couleur des cheveux et des yeux de 116 individus :

```
> tab.cont
      bleu marron vert
blond   25      6    9
brun    10     15   16
roux    12     14    9

> fisher.multcomp(tab.cont)[1]
      Pairwise comparisons by Fisher's exact test for count data

data: tab.cont

           bleu:marron bleu:vert marron:vert
blond:brun     0.02211   0.03435      0.7793
blond:roux     0.03435   0.44655      0.4801
brun:roux      0.77935   0.44655      0.5362

P value adjustment method: fdr
```

Les classes qui sont liées sont ici `blond` et `bleu`.

Test du χ^2 d'indépendance (non paramétrique)

Pour réaliser le test : `chisq.test(tab.cont)`.

Si la *p-value* du test est significative, les comparaisons deux-à-deux sont réalisées *via* `fisher.multcomp(tab.cont)`[1].

Remarque : le test du χ^2 d'indépendance est également le test de conformité à la valeur nulle du coefficient d'association de Cramér (voir fiche **83**). Ce test peut également être réalisé *via* `cramer.test(variableA,variableB)`[1] ou `cramer.test(tab.cont)`[1]. La fonction renvoie, dans l'ordre : le test de conformité à la valeur nulle (il y a « vraiment » association si le résultat est significatif), l'intervalle de confiance calculé par bootstrap et la valeur du coefficient.

Test G d'indépendance (non paramétrique)

Pour réaliser le test : `G.test(tab.cont)`[1].

Si la *p-value* du test est significative, les comparaisons deux-à-deux sont réalisées *via* `fisher.multcomp(tab.cont)`[1].

87. Analyser deux variables quantitatives interdépendantes

— PACKAGE(S) ADDITIONNEL(S) UTILISÉ(S) DANS CETTE FICHE —

[1]`RVAideMemoire`

Modèle utilisé

Le modèle utilisé est une *régression linéaire au sens des moindres rectangles*, qui diffère de la régression linéaire classique, dite au sens des moindres carrés (voie fiche **76**). En effet, dans la régression linéaire au sens des moindres carrés, une variable est considérée comme à expliquer, tandis que l'autre est considérée comme explicative. Dans la régression linéaire au sens des moindres rectangles, les deux variables sont considérées sur un même pied d'égalité, aucune n'expliquant l'autre. On dit de ces variables qu'elles sont *interdépendantes*.

Un facteur peut être ajouté à la régression pour définir des *paramètres* (*i.e.* pente et ordonnée à l'origine) différents par modalité.

Construction du modèle

Pour créer le modèle : `regression<-least.rect(variable.y~variable.x)`[1], où `variable.y` et `variable.x` sont des vecteurs contenant la valeur de chaque individu pour chaque variable (dans le même ordre). Les noms `variable.y` et `variable.x` n'ont qu'une valeur graphique : `variable.x` est destiné à être tracé en abscisses et `variable.y` en ordonnées. Les deux variables peuvent être inversées sans conséquence dans le modèle.

Pour estimer des paramètres différents selon les modalités d'un facteur : `regression<-least.rect(variable.y~variable.x|facteur)`[1], où `facteur` est un vecteur contenant la modalité de chaque individu (dans le même ordre que les deux autres variables).

Vérification de l'ajustement aux données

Avant d'aller plus loin, il est indispensable de vérifier que le modèle s'ajuste bien aux données. Cette étape est fondamentale, et ce pour tout modèle, car un test basé sur un modèle mal ajusté n'est simplement pas fiable (ni valide). Voir fiche **41** pour une explication détaillée de cette vérification.

Si le modèle est de façon flagrante non valide, transformer l'une et/ou l'autre des deux variables. Trois options sont fréquentes : $\sqrt{x}$, $log(x)$ si la première transformation n'est pas suffisante, $\sqrt[4]{x}$ ($= x^{1/4}$) en alternative à la deuxième s'il y a des 0 dans les données.

Récupération des paramètres et tests

Les valeurs des paramètres de la (ou des) régression(s) sont obtenues *via* `summary(regression)`, dans le tableau `Coefficients`. La fonction renvoie la valeur et l'intervalle de confiance de l'ordonnée à l'origine (`(Intercept)`) et de la pente (qui porte le nom de la variable x). Si un facteur a été ajouté à la régression, un tableau est renvoyé pour chacun de ces deux paramètres, contenant une ligne par modalité.

Dans le cadre de la régression linéaire au sens des moindres rectangles, on compare le plus souvent la pente à la valeur théorique 1 (qui correspond par exemple en allométrie à une relation d'isométrie entre les deux organes/structures comparés). Le résultat de ce test est également obtenu *via* `summary(regression)`. Pour changer la valeur théorique, ajouter l'argument `theo=valeur` à la fonction `least.rect()`[1]. Si un facteur a été défini dans le modèle, un test est réalisé pour chaque pente.

Il n'existe pas de test pour comparer plusieurs pentes ou ordonnées à l'origine. Cependant, lorsqu'un facteur a été défini dans le modèle, si les intervalles de confiance à 95 % des pentes (ou des ordonnées à l'origine) ne se chevauchent pas entre deux modalités, alors par définition ces pentes sont significativement différentes au seuil $\alpha = 5$ % (voir fiche **15**).

Prédiction à partir du modèle

L'intérêt d'un modèle est d'estimer les paramètres de la relation qui lie les variables x et y, mais également de prédire la valeur que prendrait la variable y pour des valeurs *connues* de la variable x (ou inversement, les deux variables étant interdépendantes). Prédire une valeur de y nécessite donc de fixer la valeur de x – et du facteur s'il y en a un dans le modèle.

Deux méthodes peuvent être utilisées pour la prédiction, les deux étant basées sur la fonction `predict()` :

— donner la valeur (ou les valeurs pour faire plusieurs prédictions) de la variable x – et éventuellement du facteur – directement dans la fonction, sous la forme d'une liste : `predict(modele,newdata=list(variables))`, où `variables` est un enchaînement de `variable.x=valeur,facteur=valeur`.

— créer un tableau contenant une colonne par chacune des deux variables connues (x et éventuellement un facteur ; les noms de colonnes doivent être *strictement identiques* aux noms des variables du modèle), et remplir chaque ligne en fonction des valeurs pour lesquelles la prédiction doit être faite (il y a donc une prédiction par ligne de ce tableau). Puis : `predict(modele,newdata=tableau)`.

EXEMPLE(S)

Avec un modèle contenant une variable x variant de 0 à 30 et un facteur à deux niveaux (A et B) :

```
> regression <- least.rect(variable.y~variable.x|facteur)[1]
```

On peut prédire une valeur de y de cette façon :

```
> predict(regression,newdata=list(variable.x=10,facteur="A"))
```

Ou, pour plusieurs prédictions :

```
> predict(regression,newdata=list(variable.x=c(10,10),facteur=c("A","B")))
```

Ou encore :

```
> predict(regression,newdata=list(variable.x=rep(10,2),facteur=c("A","B")))
```

Ou encore créer un tableau de ce type :

```
> tableau
  variable.x facteur
1         10       A
2         10       B
```

Puis :

```
> predict(regression,newdata=tableau)
```

Graphes

Illustrer la relation entre les variables x et y nécessite trois étapes :

1. Tracer les points observés : `plot(variable.y~variable.x)`.
2. Créer un vecteur ayant les mêmes minimum et maximum que la variable x mais découpé en très petits intervalles : `x <- seq2(variable.x)`[1].
3. Ajouter la droite de la relation sur le graphe, ajustée par le modèle. La détermination de tous les points de la droite est en fait basée sur une prédiction : la valeur que prend la variable y pour chaque valeur du vecteur `x`. C'est le grand nombre de valeurs de ce vecteur, et le très petit intervalle qui les sépare, qui donne l'aspect lissé de la droite (qui est en fait constituée de centaines de points reliés entre eux). Si le modèle contient un facteur, il faut en fixer toutes les valeurs à l'identique (de sorte que seule la variable x change de valeur pour toutes les prédictions). La droite s'ajoute *via* `lines(x,predict(regression,newdata=variables))` où `variables` correspond aux valeurs de la variable x et du facteur.
   Remarque : la fonction `lines()` peut être utilisée plusieurs fois de suite pour tracer plusieurs droites sur le même graphe, par exemple pour plusieurs niveaux d'un facteur.

EXEMPLE(S)

On se base toujours sur le modèle suivant :

```
> regression <- least.rect(variable.y~variable.x|facteur)[1]
```

Étape 1 : tracer les points correspondant aux données observées :

```
> plot(variable.y~variable.x)
```

Étape 2 : créer le vecteur `x` :

```
> x <- seq2(variable.x)
```

[1]

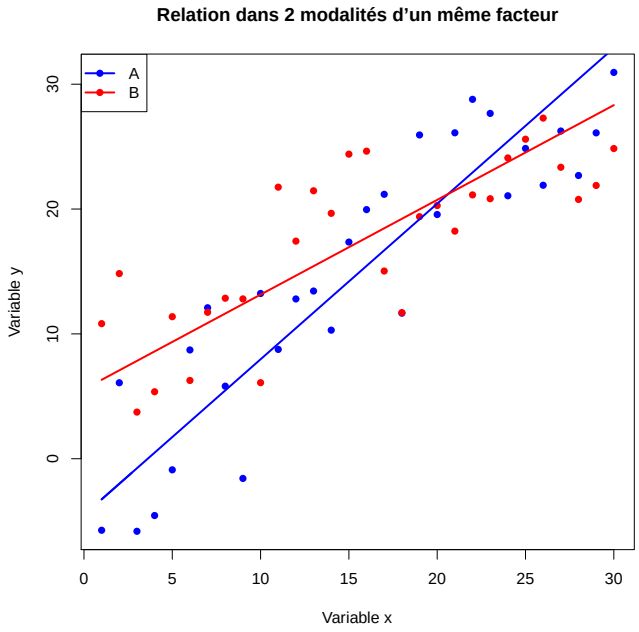
Étape 3 : ajouter la droite. On choisit de se placer dans la modalité A du facteur :

```
> lines(x,predict(regression,newdata=list(variable.x=x,facteur=rep("A",length(
x)))))
```

La fonction `rep()` est très utile dans cette situation, puisqu'elle permet de créer très simplement un vecteur de la même longueur que `x`, contenant une seule valeur répétée.

Pour ajouter la droite de la relation dans la modalité B :

```
> lines(x,predict(regression,newdata=list(variable.x=x,facteur=rep("B",length(x)
))))
```



88. Pré-traitement des données quantitatives

— PACKAGE(S) ADDITIONNEL(S) UTILISÉ(S) DANS CETTE FICHE —

[1]`Hotelling`

Avant de réaliser une analyse multivariée, il est parfois (voire souvent) bénéfique de « pré-traiter » les données. Les situations où cela peut être intéressant sont identifiées dans les fiches concernées.

De façon globale, il y a trois étapes possibles de pré-traitement (la première devant impérativement être réalisées *avant* les autres, tandis que les deux suivantes ne sont pas sensibles à leur ordre) :

Transformation : un grand nombre d'analyses multivariées sont construites sur la base d'un modèle linéaire, *i.e.* font l'hypothèses de relations linéaires entre les variables du jeu de données (voire aussi avec les variables explicatives dans les analyses à deux tableaux asymétriques). Or il arrive fréquemment que ce ne soit pas complètement le cas. De la même manière, beaucoup d'analyses font l'hypothèse que les variables ont une distribution au moins symétrique, ce qui n'est pas toujours vérifié. Enfin, beaucoup d'analyses supposent également des relations additives entre les variables, alors qu'elles peuvent être plutôt multiplicatives dans le jeu de données réel. Pour pallier à ces trois problèmes, transformer les données peut s'avérer très efficace. La transformation la plus courante et utile est le logarithme (peu importe la base de celui-ci), réalisé simplement *via* `tableau<-log(tableau)` où `tableau` est le nom du jeu de données. Une alternative au logarithme si le jeu de données contient des 0 est la racine quadratique (*i.e.* $\sqrt[4]{x} = x^{1/4}$), qui est préférable au classique $log(x+1)$ (car l'ajout d'une constante n'a pas le même effet sur les petites et les grandes valeurs). Pour la mettre en œuvre : `tableau<-tableau^(1/4)`.
Remarque : si le jeu de données est de type *compositionnel* (*i.e.* la somme des valeurs d'un individu vaut toujours 1 ou 100 %), il est *absolument indispensable* de transformer les données avant toute analyse multivariée. La transformation la plus courante dans ce cas est celle du *Centered LogRatio* (CLR), réalisée *via* : `tableau<-clr(tableau)`[1]. Si le jeu de données contient des 0 il n'y a pas d'alternative à ajouter une constante à toutes les valeurs ; la choisir pour qu'elle soit d'un ordre de grandeur plus faible que la plus petite valeur non nulle du tableau.

Centrage : le centrage consiste à *soustraire* à chaque valeur du tableau la moyenne de la colonne dans laquelle cette valeur se trouve (la conséquence est que toutes les colonnes ont ensuite une moyenne nulle). Cela permet de s'affranchir de l'ordre de grandeur des différentes variables, ce qui est d'autant plus intéressant que ces ordres de grandeur sont différents. Pour les analyses basées sur une matrice de distance (*i.e.* toute classification ou les ordinations sur une matrice de distance), l'effet est potentiellement très important (mais bénéfique – sauf à vouloir tenir compte des différences d'ordre de grandeur dans l'analyse). Pour les méthodes d'ordination basées sur un tableau de variables, le centrage n'a aucun effet à part simplifier les calculs internes.
Remarque : plus généralement, centrer les données consiste à soustraire toute valeur, du moment qu'elle est constante par colonne. Les cas où ce n'est pas la moyenne qui est utilisée sont très rares, et reposent sur des hypothèses bien particulières.

Réduction : la réduction consiste à *diviser* chaque valeur du tableau par une constante propre à chaque colonne. Il existe beaucoup de méthodes de réduction, dont les objectifs ne sont pas toujours identiques. Globalement, les méthodes les plus courantes visent à réduire la différence de variabilité entre les variables. L'intérêt est d'équilibrer le poids donné à chaque variable dans l'analyse, car dans de nombreuses méthodes ce poids est fonction de la variabilité. La méthode la plus fréquente est de diviser par l'écart-type de la colonne, ce qui ramène toutes les variables à une variance de 1. Autrement dit, on supprime tout *a priori* dans l'analyse pour donner exactement le même poids à toutes les variables. Sauf hypothèse particulière, ce type de réduction est le plus souvent très bénéfique.

Pour centrer (sur la base de la moyenne) et réduire (sur la base de l'écart-type) un tableau : `tableau<-scale(tableau)`. Voir `?scale` pour d'autres manières de centrer et réduire.

L'expression « tableau centré-réduit » sous-ented un centrage par la moyenne et une réduction par l'écart-type. C'est aussi ce qui est sous-entendu dans « tableau standardisé » ou « normalisé ».

89. Interpréter un cercle des corrélations

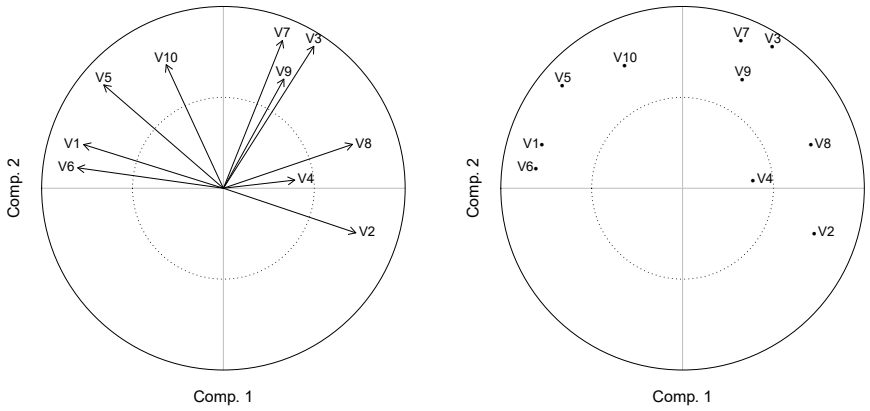
Le cercle des corrélation est une figure classique pour interpréter les résultats d'un grand nombre d'analyses multivariées. Il permet de montrer à la fois la corrélation entre les variables (*i.e.* les flèches) et les axes de l'analyse, et la corrélation entre les variables elles-mêmes.

Son interprétation repose sur trois principes simples :

— Plus une flèche est longue, mieux l'information portée par la variable est bien synthétisée par les deux axes représentés. Pour l'interprétation, on se concentre donc sur les flèches les plus longues (*i.e.* les variables les mieux représentées).
— L'angle entre deux flèches (ou entre une flèche et un axe) indique la corrélation entre les deux variables (ou entre une variable et un axe) :
  — angle aigu = positive (0 ° = corrélation 1)
  — angle droit = nulle (90 ° = corrélation 0)
  — angle obtus = corrélation négative (180 ° = corrélation -1)
— Plus un individu est situé vers l'avant d'une flèche (quand on le projette perpendiculairement à cette flèche) et plus sa valeur pour cette variable est élevée (et vice-versa).

Lorsque l'on met en parallèle un graphe des individus et un cercle des corrélations, on peut ainsi identifier les variables qui structurent ces individus, voire interpréter biologiquement les axes.

Remarque : parfois les variables ne sont pas représentées par des flèches mais par des points sur le cercle des corrélations. Cela ne change strictement rien à l'interprétation, car ces points correspondent à l'extrémité des flèches. Utiliser des points peut rendre le graphe plus lisible si les variables sont nombreuses, mais les angles entre variables (ou entre variables et axes) n'apparaissent plus aussi clairement.



90. Utiliser les axes d'une ordination comme variables d'une autre analyse

— PACKAGE(S) ADDITIONNEL(S) UTILISÉ(S) DANS CETTE FICHE —

`RVAideMemoire`[1]

Il est toujours possible d'utiliser les axes d'une ordination (ACP, PCoA, analyse mixte…) comme variables à expliquer dans une autre analyse. Cela peut permettre :

- de diminuer drastiquement le nombre de variables (puisque les premiers axes d'une analyse multivariée capturent l'essentiel de l'information, quand une synthèse efficace est possible)
- de supprimer la corrélation entre les variables originales (puisqu'un des principes des analyses multivariées est que leurs axes ne sont pas – ou très peu – corrélés, *i.e.* ils apportent tous une information non captée par les autres axes)
- de passer de variables qualitatives à des variables quantitatives
- de passer d'une matrice de distance à un tableau de variables
- …

En pratique on peut ainsi chercher à tester la séparation de plusieurs groupes sur un ou deux axes (voir fiche **91**), à synthétiser un ensemble de variables par une seule variable de synthèse à laquelle toutes les autres sont corrélées, à comparer le résultat de plusieurs ordinations… Les possibilités sont nombreuses.

Remarque 1 : si les coordonnées des individus sur un axe d'une nMDS sont utilisés dans une analyse univariée, celle-ci doit être non paramétrique car la nMDS n'utilise qu'une information semi-quantitative (*i.e.* relative, voir fiche **103**). Pour la même raison, la plupart des analyses à plusieurs tableaux ne sont pas pertinentes avec la nMDS (excepté l'analyse procustéenne, voir fiche **113**).

Remarque 2 : les coordonnées des individus sur les axes d'une PLS-DA *ne doivent pas* être utilisés pour tester la séparation de plusieurs groupes. Un test dédié doit être utilisé (voir fiche **106**).

Récupération des coordonnées des individus

Pour toutes les analyses multivariées présentées dans ce document (et bien d'autres), on peut récupérer les coordonnées des individus sur un ou plusieurs axes *via* `MVA.scores(analyse)`[1] où `analyse` est le nom de l'analyse multivariée (les coordonées sont dans le compartiment `$coord` de la liste renvoyée par la fonction). Les arguments `xax` et `yax` permettent de préciser sur quels axes les coordonnées doivent être récupérées ; ils valent 1 et 2 par défaut, respectivement. Si plus de deux axes sont souhaités, récupérer les coordonnées en spécifiant tous les axes à l'argument `xax`.

— EXEMPLE(S) —

On souhaite récupérer les coordonnées des individus sur les deux premiers axes d'une ACP (nommée `ACP`) :

```
> MVA.scores(ACP)$coord[1]
```

Pour récupérer les coordonnées sur les axes 1 et 3 :

```
> MVA.scores(ACP,xax=1,yax=3)$coord[1]
```

ou :

```
> MVA.scores(ACP,xax=c(1,3))$coord[1]
```

Pour récupérer les coordonnées sur les trois premiers axes :

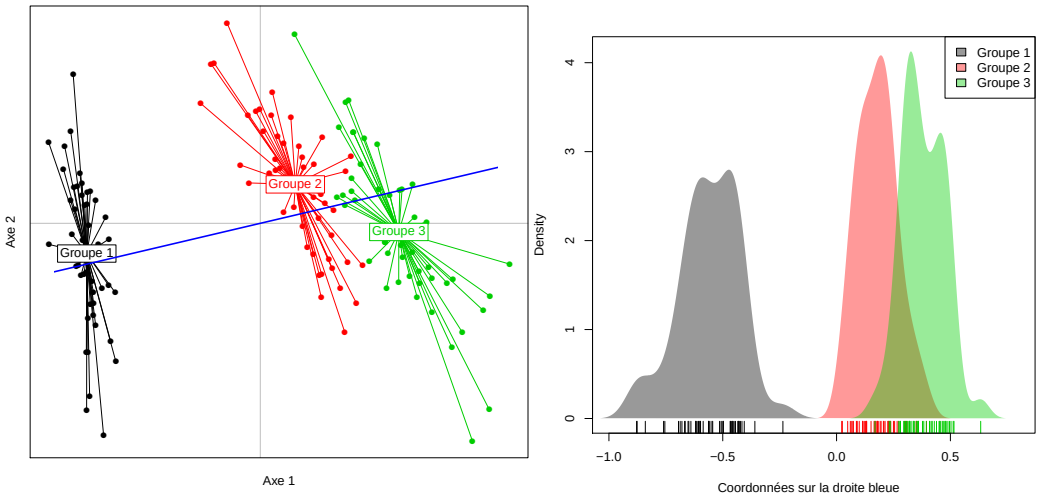
```
> MVA.scores(ACP,xax=1:3)$coord[1]
```

Attention, certaines analyses renvoient des coordonnées à la fois pour les lignes et les colonnes du tableau de départ (l'AFC (voir fiche **98**) et la CCA (voir fiche **107**)), tandis que d'autres renvoient les coordonnées des individus dans plusieurs espaces multivariés (la plupart des analyses à deux ou plus de deux tableaux). Les arguments `set` et `space` de la fonction `MVA.scores()`[1] permettent de préciser quelles coordonnées récupérer, voir `?MVA.scores` pour des informations détaillées.

Remarque : attention également à l'argument `scaling` pour l'AFC et la CCA (voir fiches **98** et **107**).

Il peut arriver que l'on veuille récupérer les coordonnées des individus dans une autre direction qu'un axe (*i.e.* une diagonale entre deux axes), car c'est celle-ci qui est la plus pertinente pour l'analyse ultérieure. La procédure se fait alors en plusieurs étapes :

1. Tracer le graphe sur le plan factoriel d'intérêt grâce à la fonction `MVA.plot()`[1] (voir les fiches des différentes analyses multivariées pour plus de détail).
2. Créer l'objet `droite<-loc.slp()`[1]. La fonction `loc.slp()`[1] (qui n'accepte aucun argument) demande alors de cliquer deux fois sur le graphe, pour définir la direction d'intérêt. On peut ensuite afficher cette direction *via* `abline(0,droite)`, et éventuellement recommencer l'opération pour modifier la droite (en relançant `loc.slp()`[1]).
3. Récupérer les coordonnées des individus sur les deux axes du plan factoriel grâce à la fonction `MVA.scores()`[1].
4. Calculer les coordonnées des individus une fois projetés sur la droite définie à l'étape 2 : `coord.proj(coord,droite)`[1] où `coord` est le tableau des coordonnées récupérées à l'étape 3.



Cas particulier de l'analyse de co-inertie

L'analyse de co-inertie (voir fiche **114**) est réalisée par la fonction `coinertia()` du package `ade4`. Cette fonction a la particularité de n'accepter en arguments que des ordinations (pas des tableaux de variables), qui plus est réalisées grâce à des fonctions de ce même package. En pratique les arguments sont des ordinations à un seul tableau : ACP, AFC, ACM ou analyse mixte. Si l'ACM (voir fiche **99**) et l'analyse mixte (voir fiche **100**) sont bien réalisées grâce au package `ade4`, l'ACP (voir fiche **97**) et l'AFC (voir fiche **98**) sont réalisées grâce au package `vegan`. Pour convertir le résultat d'une de ces ordinations en objet utilisable par la fonction `coinertia()` : `analyse2<-to.dudi(analyse1)`[1] où `analyse1` est le nom de l'ACP, AFC ou PCoA.

91. Relation entre une ordination et des variables externes

— PACKAGE(S) ADDITIONNEL(S) UTILISÉ(S) DANS CETTE FICHE —

[1]`vegan`, [2]`RVAideMemoire`

Il est toujours possible de tester si le résultat d'une ordination est « corrélé » à des variables externes, *i.e.* non prises en compte dans l'analyse. Plus précisément, le test a toujours lieu dans un *plan factoriel* donné (*i.e.* un plan formé de deux axes) ou sur une seule dimension (un axe ou n'importe quelle diagonale). La ou les variable(s) considérée(s) dans le test est (sont) donc les coordonnées des individus dans le plan factoriel, ou sur une unique droite. Les variables externes peuvent être quantitatives et/ou qualitatives.

En général ce genre d'approche concerne des analyses à un tableau (ACP, AFC, PCoA...), mais il y a peu de restriction *a priori*. La PLS-DA est une exception, puisque la séparation entre les groupes *ne doit pas* être testée de cette manière mais grâce à un test dédié (voir fiche **106**).

Remarque : la nMDS est une analyse multivariée qui n'utilise qu'une information semi-quantitative (*i.e.* relative, voir fiche **103**). La seule approche réellement pertinente pour tester la relation avec des variables externes est donc non paramétrique, ce qui oblige à ne travailler que dans une seule dimension (et pas un plan factoriel, car les tests applicables en deux dimensions sont paramétriques).

Test sur une dimension

Commencer par récupérer les coordonnées des points sur la dimension choisie (un axe ou n'importe quelle diagonale, voir fiche **90**). La nouvelle variable ainsi créée, qui est quantitative continue non bornée, peut ensuite être utilisée dans n'importe quelle analyse univariée ou bivariée.

Test sur deux dimensions

Pour réaliser le test (non paramétrique) : `envfit(formule)`[1]. Voir fiche **40** pour une explication détaillée de la construction d'une formule. Dans cette formule, la réponse est soit un tableau contenant les coordonnées des individus sur les deux axes à tester (voir fiche **90** pour les récupérer), soit plus simplement une ordination si elle a été créée avec le package `vegan`.

Remarque : ce test ne prend pas en compte les interactions entre variables externes.

Variables externes qualitatives

Si une variable qualitative a un effet significatif et qu'elle comporte plus de deux modalités, il est nécessaire de réaliser des comparaisons multiples pour identifier les modalités qui diffèrent. Utiliser pour cela la fonction `pairwise.factorfit(reponse,facteur)`[2] où `reponse` est l'ordination et `facteur` le facteur d'intérêt.

Pour la représentation graphique, les fiches des différentes analyses multivariées expliquent comment ajouter les modalités d'un facteur sur un plan factoriel.

Variables externes quantatives

Dans le cas d'une variable quantitative, on peut voir de quelle manière elle corrèle à l'ordination *via* une procédure en trois (ou quatre) étapes :

1. Tracer le graphe sur le plan factoriel d'intérêt grâce à la fonction `MVA.plot()`[2] (voir les fiches des différentes analyses multivariées pour plus de détail).
2. Stocker le résultat de la fonction `envfit()`[1] dans un objet : `test<-envfit(formule)`[1].
3. Si le test réalisé à l'étape précédente incluait des variables qualitatives : `test$factors<-NULL`.
4. Ajouter l'information des variables externes sur le graphe : `plot(test)`. La longueur des flèches est arbitraire mais l'interprétation en termes de corrélation est la même que pour un cercle des corrélations (voir fiche **89**).

Remarque : la procédure est identique pour ajouter des variables externes quantitatives sur un cercle des corrélations ; seule la première étape est différente. La longueur des flèches des variables externes est là aussi arbitraire, mais l'interprétation en termes de corrélation est valide.

92. Classification – Identifier et tester une hiérarchie

— PACKAGE(S) ADDITIONNEL(S) UTILISÉ(S) DANS CETTE FICHE —

[1]ape, [2]pvclust

La hiérarchie est obtenue grâce à une méthode de classification dite… *hiérarchique*, qui aboutit à un arbre (de type phylogénétique) appelé *dendrogramme*. Pour obtenir un dendrogramme, il y a deux possibilités :

— Partir des individus et les agglomérer petit à petit jusqu'à ne former qu'un seul groupe les réunissant tous. C'est l'objet des méthodes *ascendantes* hiérarchiques.

— Partir d'un unique groupe réunissant tous les individus et le diviser petit à petit jusqu'à former autant de groupes qu'il y a d'individus. C'est l'objet des méthodes *descendantes* hiérarchiques.

Les méthodes ascendantes donnent globalement de meilleurs résultats que les méthodes descendantes. Dans cette fiche est donc uniquement présentée la Classification Ascendante Hiérarchique (CAH).

Le dendrogramme est créé *via* `dendro<-hclust(mat.dist,method="ward.D2")`, où `mat.dist` est une matrice de distance (calculée à partir d'un ensemble de variables ou qui constitue le jeu de données lui-même, voir fiche **101**). L'argument `method` précise le critère sur lequel se fait le regroupement des individus, celui de Ward est le plus courant. Les autres méthodes usuelles sont l'UPGMA (`method="average"`) et le WPGMA (`method="mcquitty"`).

Pour visualier le dendrogramme : `plot(dendro,hang=-1)`.

Si plusieurs dendrogrammes ont été obtenus à partir des mêmes données (par exemple en utilisant des critères de regroupement différents) et qu'ils diffèrent, on peut chercher à identifier un *arbre consensus*. Pour cela : `cons<-consensus(list(as.phylo(dend1),as.phylo(dend2),...))`[1] où `dend1`, `dend2` et `...` sont les dendrogrammes. Par défaut la fonction `consensus()`[1] ne retient que les groupes qui apparaissent dans 100 % des dendrogrammes. Ce pourcentage peut être diminué, jusqu'à 50 %, grâce à l'argument `p` (qui doit valoir entre 0,5 et 1).

Pour visualier l'arbre consensus : `plot(cons)`.

À la condition que `mat.dist` ait été obtenue à partir d'un tableau de variables, on peut tester la solidité des groupes générés par la CAH. On utilise pour cela une technique de *bootstrap* : `dendro.boot<-pvclust(t(tab),method.dist=FONCTION,,method.hclust=critere,quiet=TRUE)`[2], où `tab` est le tableau de données et `critere` le critère sur lequel se fait le regroupement des individus dans la CAH (forcément une méthode acceptée par la fonction `hclust()` : `"ward.D2"`, `"average"` ou `"mcquitty"` dans l'essentiel des cas).

L'argument `FONCTION`, qui indique la manière dont est calculée la matrice de distance à partir du tableau de variables (voir fiche **101**), peut être de deux natures :

— soit c'est une méthode proposée par la fonction `dist()`, auquel cas on peut simplement donner le nom de cette méthode entre guillemets (voir `?dist` pour la liste complète)

— soit ce n'est pas une méthode proposée par la fonction `dist()`, auquel cas la syntaxe est plus complexe et de la forme `function(x) {f(t(x),arguments)}` où `f` est le nom de la fonction calculant la matrice de distance et `arguments` ses éventuels arguments (voir fiche **101**).

Remarque 1 : la fonction `pvclust()`[2] nécessite un temps de calcul qui augmente assez rapidement avec la taille du jeu de données. Si le package `parallel` est installé, on peut accélérer ce temps de calcul en ajoutant l'argument `parallel=TRUE`.

Remarque 2 : à ce jour la fonction `pvclust()`[2] n'accepte aucun tableau de données contenant le moindre facteur (toutes les variables doivent être quantitatives).

EXEMPLE(S)

On veut appliquer la technique du *bootstrap* à une CAH réalisée sur le tableau de variables `tab`. La distance utilisée est la distance Euclidienne (voir fiche **101**), et le regroupement des individus se fait grâce au critère de Ward :

```
> dendro.boot <- pvclust(t(tab),method.dist="euclidean",method.hclust="ward.D2",
quiet=TRUE)
```
[2]

On veut cette fois utiliser la distance d'Hellinger (voir fiche **101**), et regrouper les individus par la méthode UPGMA :

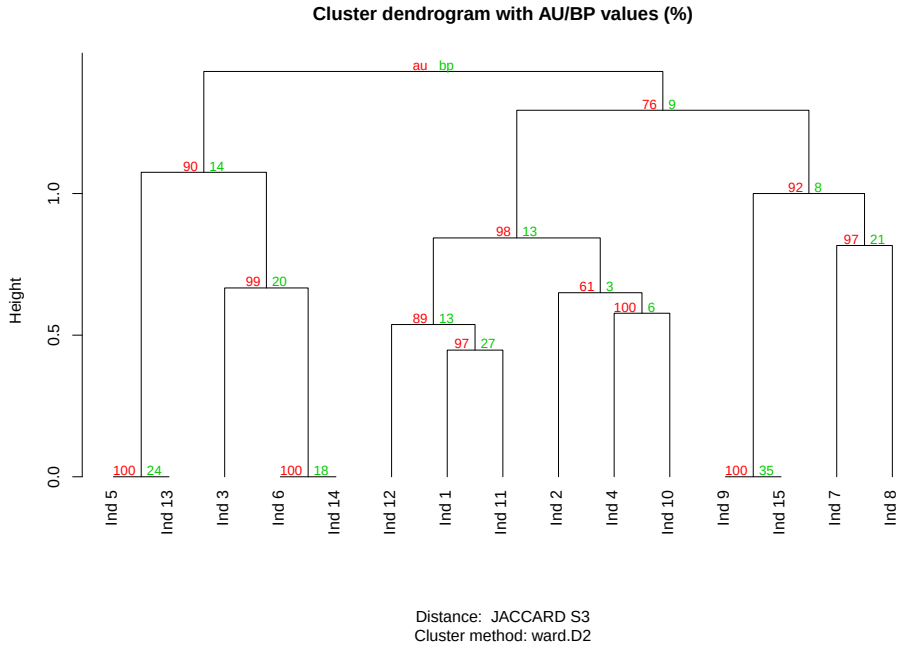
```
> dendro.boot <- pvclust(t(tab),method.dist=function(x) {vegdist(decostand(t(x),
method="hellinger"),method="euclidean")},method.hclust="average",quiet=TRUE)}
```
[2]

Avec une distance de Jaccard (voir fiche **101**) et un regroupement par la méthode WPGMA :

```
> dendro.boot <- pvclust(t(tab),method.dist=function(x) {dist.binary(t(x),meth-
od=1)},method.hclust="mcquitty",quiet=TRUE)}
```
[2]

Pour représenter les résultats du *bootstrap* : `plot(dendro.boot,hang=-1,print.num=FALSE)`. Le dendrogramme est identique à celui produit par la fonction `hclust()`, mais la fonction `pvclust()`[2] ajoute deux valeurs au-dessus du point de départ de chaque groupe : l'*Approximately Unbiased p-value* (*AU value*) en rouge, et la *Bootstrap Probability value* (*BP value*) en vert. Ce sont deux valeurs qui varient entre 0 et 100, et indiquent la solidité d'un groupe. On considère généralement qu'une valeur ≥ 95 indique un groupe solide (« significatif »). N'utiliser que l'*AU value* pour conclure, car la *BP value* est biaisée.

Une aide à l'interprétation graphique peut être apportée *via* `pvrect(dendro.boot,max.only=FALSE)`[2]. La fonction ajoute au graphe créé à l'étape précédente des rectangles montrant tous les groupes avec une *AU value* ≥ 95 (utiliser l'argument `alpha` pour modifier ce seuil, en le faisant varier entre 0 et 1 – 0,95 par défaut).



93. Classification – 1. Tester si un jeu de données peut être classifié

— PACKAGE(S) ADDITIONNEL(S) UTILISÉ(S) DANS CETTE FICHE —

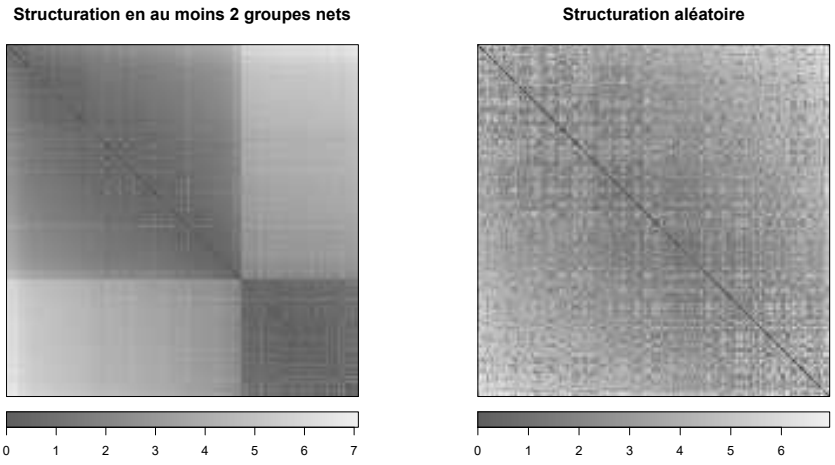
[1]`seriation`, [2]`clustertend`

Une méthode de classification, quelle qu'elle soit, donnera toujours un résultat (*i.e.* la constitution de groupes). Et ce même si aucun groupe n'existe réellement dans les données, *i.e.* même si la structuration des individus est complètement aléatoire. Avant de réaliser une classification, il est donc essentiel de s'assurer qu'il existe bien une certaine structuration dans le jeu de données (en anglais *clustering tendency*).

Deux méthodes complémentaires peuvent être employées pour tester s'il existe une structuration, l'une graphique et l'autre statistique.

Méthode graphique

La méthode est appelée *Visual Assessment of cluster Tendency* (*VAT*). Pour la mettre en œuvre : `dissplot(mat.dist)`[1] où `mat.dist` est une matrice de distance (calculée à partir d'un ensemble de variables ou qui constitue le jeu de données lui-même, voir fiche **101**). La fonction renvoie un graphe représentatnt la matrice de distance, où les individus ont été automatiquement ré-ordonnés pour regrouper les plus semblables. L'interprétation est simple : si au moins deux carrés foncés apparaissent le long de la diagonale, il existe des groupes dans le jeu de données (mais peu importe le nombre à ce stade) ; si aucune structuration n'apparaît, les individus sont structurés aléatoirement. Dans le premier cas on peut utiliser une procédure de classification pour révéler les groupes, dans le second cas l'analyse s'arrête ici.



Méthode statistique

La méthode ne peut être utilisée que sur un tableau de variables, et repose sur la statistique d'Hopkins. Une valeur de 0.5 de cette statistique indique une structuration aléatoire des individus, tandis qu'une valeur qui s'en éloigne pour se rapprocher de 0 indique qu'il existe une structuration.

Pour calculer la statistique d'Hopkins : `hopkins(tableau,n=10)`[2] où `tableau` est le jeu de données. L'argument `n` est un nombre nécessaire à l'algorithme de calcul (il peut valoir au maximum le nombre de lignes de `tableau` −1).

94. Classification – 2. Identifier le nombre optimal de groupes

— PACKAGE(S) ADDITIONNEL(S) UTILISÉ(S) DANS CETTE FICHE —

[1]`NbClust`

⚠ Il est supposé que l'étape précédente d'une bonne classification a été réalisée (voir fiche **93**). ⚠

Identifier le nombre optimal de groupes est une des étapes les plus importantes d'une procédure de classification. La stratégie repose sur l'utilisation d'un *indice*, que l'on calcule pour toute une série de nombres de groupes possibles (2, 3, 4…). Le nombre de groupes qui donne la meilleure valeur de l'indice indique le nombre de groupes optimal.

Il n'existe cependant pas qu'un indice, mais au moins une trentaine qui ne renvoient pas toujours le même nombre de groupes optimal. L'idée est donc de calculer tous ces indices, et de retenir comme nombre de groupes optimal la valeur renvoyée le plus fréquemment. La procédure diffère légèrement selon que le jeu de données est un tableau de variables ou directement une matrice de distance (voir fiche **101**).

Le jeu de données est un tableau de variables

Pour mettre en œuvre la procédure : `nc<-NbClust(tableau,diss=mat.dist,distance=NULL,method="ward.D2")`[1] où `tableau` est le jeu de données et `mat.dist` la matrice de distance calculée à partir de ce même jeu de données (voir fiche **101**). Le résultat de la fonction est socké dans un objet (ici `nc` mais peu importe son nom) qui ne sera pas réutilisé par la suite, cela simplifie seulement l'affichage des résultats. Le nombre de groupes minimal testé est 2, et le maximal 15. Pour changer ces valeurs utiliser les arguments `min.nc` et `max.nc`.

La fonction calcule la valeur de 26 indices pour 2 à 15 groupes (par défaut) et renvoie deux sorties complémentaires :

— Un résumé des nombres de groupes optimaux calculés (la somme peut parfois être inférieure à 26). Le nombre le plus fréquent est retenu comme optimal.
— Deux graphes à propos d'un *Dindex*. Sur celui de droite, la valeur la plus élevée indique le nombre de groupes optimal.

Le jeu de données est une matrice de distance

Seuls cinq indices sont calculables et il faut effectuer l'opération pour chacun d'entre eux. Pour cela : `nc<-NbClust(diss=mat.dist,distance=NULL,method="ward.D2",index=indice)`[1] où `indice` vaut successivement `"frey"`, `"mcclain"`, `"cindex"`, `"silhouette"` et `"dunn"`. À chaque fois, noter le nombre de groupes optimal présent dans le compartiment `$Best.nc` de la liste renvoyée, sous l'intitulé `Number_clusters`. Considérer comme nombre de groupes optimal la valeur renvoyée le plus fréquemment parmi les cinq indices.

95. Classification – 3. Réaliser la classification

— PACKAGE(S) ADDITIONNEL(S) UTILISÉ(S) DANS CETTE FICHE —

[1]`cluster`, [2]`factoextra`, [3]`clValid`, [4]`fpc`

⚠ Il est supposé que les étapes précédentes d'une bonne classification ont été réalisées (voir fiches **93** à **94**). ⚠

Il existe deux grandes familles de méthodes de classification :

— Les méthodes de *partitionnement*, auxquelles on doit fournir le nombre de groupes *a priori* (identifié par des analyses dédiées, voir fiche **94**) et qui trouvent la meilleure façon de créer de tels groupes.

— Les méthodes *hiérarchiques*, qui n'ont pas d'*a priori* sur le nombre de groupes et aboutissent à un arbre (de type phylogénétique) appelé *dendrogramme*. Cet arbre est ensuite « coupé » au niveau du nombre de groupes optimal (voir fiche **94**). Pour obtenir un dendrogramme, il y a deux possibilités :

  — Partir des individus et les agglomérer petit à petit jusqu'à ne former qu'un seul groupe les réunissant tous. C'est l'objet des méthodes *ascendantes* hiérarchiques.

  — Partir d'un unique groupe réunissant tous les individus et le diviser petit à petit jusqu'à former autant de groupes qu'il y a d'individus. C'est l'objet des méthodes *descendantes* hiérarchiques.

Partitionnement

La méthode décrite ici est celle des *k-medoids* (ou *Partitioning Around Medoids, PAM*), une extension plus robuste de la méthode plus connue des *k-means*.

Pour la mettre en œuvre : `classif<-pam(mat.dist,k=nb)`[1] où `mat.dist` est une matrice de distance (calculée à partir d'un ensemble de variables ou qui constitue le jeu de données lui-même, voir fiche **101**) et `nb` le nombre de groupes souhaité. Le groupe auquel appartient chaque individu est renvoyé dans `classif$clustering`.

Pour les grands jeux de données, voir `?clara`[1].

Classification hiérarchique

Classification ascendante hiérarchique

La première étape est de créer le dendrogramme : `dendro<-agnes(mat.dist,method="ward")`[1]. L'argument `method` précise le critère sur lequel se fait le regroupement des individus, celui de Ward est le plus courant. Les autres méthodes usuelles sont l'UPGMA (`method="average"`) et le WPGMA (`method="weighted"`). Pour visualiser le dendrogramme : `fviz_dend(dendro)`[2].

La seconde étape est de couper le dendrogramme au niveau adéquat pour obtenir le nombre optimal de groupes. Pour cela : `classif<-cutree(dendro,k=nb)`. La fonction renvoie le groupe auquel appartient chaque individu. Pour représenter ces groupes sur le dendrogramme, ajouter l'argument `k=nb` à la fonction `fviz_dend()`[2].

Classification descendante hiérarchique

Les deux étapes sont similaires à la méthode précédente, sauf que le dendrogramme est construit dans le sens inverse. On créé d'abord le dendrogramme : `dendro<-diana(mat.dist)`[1]. Pour le visualiser : `fviz_dend(dendro)`[2]. On coupe ensuite le dendrogramme pour affecter chaque individu à un groupe : `classif<-cutree(dendro,k=nb)`. On peut là encore représenter les groupes sur le dendrogramme en utilisant l'argument `k=nb` de la fonction `fviz_dend()`[2].

Quelle méthode choisir ?

Aux conditions (i) que le jeu de données soit un tableau de variables et (ii) que la distance utilisée pour calculer la matrice de distance soit euclidienne, de Manhattan ou basée sur la corrélation de Pearson (voir fiche **101**), on peut directement comparer le résultat des trois méthodes dans une procédure en deux

étapes. D'abord `test<-clValid(tableau,nClust=nb,clMethods=c("pam","agnes","diana"), validation=c("internal","stability"),method="ward")`[3] où `tableau` est le jeu de données et `nb` le nombre de groupes optimal. L'argument `validation` précise que l'on veut comparer les trois méthodes à la fois sur des critères dits *internes* et d'autres dits de *stabilité*. Le résultat de la comparaison est ensuite obtenu *via* `summary(test)`, dans le tableau `Optimal Scores`. Pour chacun des sept critères (les quatre premiers de stabilité, les trois suivants internes), la colonne `Method` donne la méthode la plus efficace. Choisir celle qui ressort le plus souvent comme la plus efficace.

Remarque : la fonction `clValid()`[3] utilise la distance euclidienne par défaut. Utiliser l'argument `metric` pour choisir une autre mesure de distance.

Si l'on ne peut pas réaliser directement la comparaison des trois méthodes, on peut toujours le faire manuellement. On se base pour cela sur trois indices de validation interne : la largeur de Silhouette (*Silhouette width*) (voir fiche **95**), l'indice de Dunn et la connectivité. Pour chacune des trois méthodes de classification, calculer ces indices :

— largeur de Silhouette : `cluster.stats(mat.dist,clustering=classif)$avg.silwidth`[4] où `classif` est le vecteur contenant le groupe de chaque individu (sous forme numérique)
— indice de Dunn : `cluster.stats(mat.dist,clustering=classif)$dunn`[4] (idem)
— connectivité : `connectivity(mat.dist,clusters=classif)`[3] (idem).

Du point de vue de la largeur de Silhouette, la méthode donnant la valeur maximale est la meilleure. Il en est de même pour l'indice de Dunn. Du point de vue de la connectivité, la méthode donnant la valeur minimale est la meilleure.

96. Classification – 4. Valider le résultat d'une classification

— PACKAGE(S) ADDITIONNEL(S) UTILISÉ(S) DANS CETTE FICHE —

[1]`cluster`

⚠ Il est supposé que les étapes précédentes d'une bonne classification ont été réalisées (voir fiches **93** à **95**). ⚠

Les méthodes de classification sont globalement très efficaces, mais il peut arriver qu'un petit nombre d'individus soient mal classés (ou du moins c'est ce que l'on suppose, puisqu'on ne connaît pas les « vrais » groupes). Pour identifier ces individus, on peut utiliser l'indice Silhouette. Celui-ci est calculé pour chaque individu et varie entre -1 et 1 :

— une valeur proche de 1 indique un individu très probablement bien classé
— une valeur proche de 0 indique un individu « à cheval » entre deux groupes
— une valeur négative indique un individu probablement mal classé.

Pour calculer les valeurs de l'indice Silhouette : `sil<-silhouette(classif,mat.dist)`[1] où `classif` est le vecteur contenant le groupe de chaque individu (sous forme numérique, voir fiche **95**) et `mat.dist` la matrice de distance (calculée à partir d'un ensemble de variables ou qui constitue le jeu de données lui-même, voir fiche **101**). La fonction renvoie un tableau avec pour chaque individu son groupe tel que défini par la méthode de classification (colonne `cluster`), le groupe voisin dont il est le plus proche (colonne `neighbor`) et l'indice Silhouette (colonne `sil_width`). Si un individu a une valeur d'indice négative, changer manuellement son groupe dans le vecteur `classif` en le remplaçant par le groupe le plus proche (auquel il appartient plus probablement).

Pour aider à s'y retrouver dans les valeurs d'indice Silhouette, on peut :

— les représenter graphiquement : `plot(sil)`. Sur le graphe sont aussi affichés les effectifs par groupe, l'indice Silhouette moyen par groupe et l'indice Silhouette moyen tous groupes confondus
— les réordonner dans le même ordre que sur le graphe : `sortSilhouette(sil)`[1].

— EXEMPLE(S) —

On obtient les valeurs suivantes d'indice Silhouette (uniquement les cinq premiers individus) :

```
     cluster neighbor    silwidth
[1,]       3        2  0.46165580
[2,]       2        3  0.17075888
[3,]       2        3 -0.04842929
[4,]       3        2  0.55999150
[5,]       3        2  0.46112097
```

L'individu 3, qui avait été classé dans le groupe 2, a une valeur d'indice négative. Il appartient donc probablement non pas au groupe 2 mais au groupe le plus proche, *i.e.* le groupe 3. On change donc le groupe de cet individu dans le vecteur `classif` (qui contient les groupes) :

```
> classif[3] <- 3
```

97. L'analyse en composantes principales

— PACKAGE(S) ADDITIONNEL(S) UTILISÉ(S) DANS CETTE FICHE —

[1]`vegan`, [2]`RVAideMemoire`

SYNONYMES, TRADUCTIONS ET ABRÉVIATIONS

Français – Analyse en composantes principales (ACP)
Anglais – *Principal component analysis* (*PCA*)

Préparation des données

L'ACP fonctionne d'autant mieux que les variables ont une distribution à peu près normale (au moins symétrique) et qu'elles sont reliées entre elles par des relations linéaires. Une transformation préalable des données peut grandement aider à améliorer la situation (voir fiche **88**).

Il est également recommandé, la plupart du temps, de standardiser les variables avant l'analyse (voir fiche **88**). Cela permet de donner le même poids à toutes les variables, et d'interpréter les résultats en termes de corrélation ce qui est souvent plus facile. Dans cette fiche on considèrera que les variables sont standardisées.

Réalisation de l'analyse

Pour réaliser l'ACP : `ACP<-rda(tableau)`[1] où `tableau` est le tableau de données. Si l'on souhaite standardiser les variables mais que l'on n'a pas effectué l'opération au préalable, ajouter l'argument `scale=TRUE`. Par défaut les variables *ne sont pas standardisées*.

Qualité de l'analyse

L'ACP a pour objectif de synthétiser au mieux une certaine information qui est la *variance totale* du jeu de données. Pour estimer la qualité de l'analyse, on s'intéresse donc au pourcentage de variance expliqué par chaque axe. Pour obtenir ces pourcentages : `MVA.synt(ACP)`[2].

Remarque 1 : les pourcentages de variance sont toujours en ordre décroissant (*i.e.* l'axe 1 explique plus de variance que l'axe 2, qui en explique lui-même plus que l'axe 3...).

Remarque 2 : il n'y a pas de règle absolue sur le nombre d'axes à retenir pour l'interprétation. Il s'agit toujours d'un compromis entre une bonne synthèse de l'information du jeu de données (qui augmente avec le nombre d'axes) et une facilité à interpréter (qui diminue avec le nombre d'axes).

Il peut arriver que les pourcentages de variance expliqués par l'ACP soient relativement faibles. Cela ne veut pas forcément dire que l'analyse est inutile. En effet, ce qui compte est que les distances interindividuelles dans l'espace multivarié créé par l'analyse soient bien représentatives des distances interindividuelles réelles (*i.e.* dans le tableau de données). Pour vérifier cela, on trace un *diagramme de Shepard* : `stressplot(ACP)`[1]. Sur ce diagramme, si les points sont à peu près alignés le long d'une droite alors les distances dans l'espace de l'ACP sont bien proportionnelles aux distances réelles, et l'on peut se baser sur les résultats de l'analyse pour l'interprétation. Si les points ne dessinent clairement pas une droite, les distances ne sont pas préservées et interpréter l'analyse est inutile car elle ne représente pas la réalité.

Remarque 1 : la droite rouge du diagramme de Shepard indique une proportionnalité parfaite entre les distances. Si les points forment une droite proche de la droite rouge cela est signe d'un pourcentage de variance expliqué élevé, s'ils forment une droite plus éloignée que ce pourcentage est moins important (mais tout de même qu'il y a proportionnalité des distances).

Remarque 2 : par défaut le diagramme de Shepard considère les deux premiers axes de l'ACP. Si l'interprétation est basée sur plus d'axes, ajouter l'argument `k=nb` où `nb` est le nombre total d'axes utilisés.

Représentations graphiques

En ACP on a deux représentations possibles : le *graphe des individus* qui montre la position de ces individus sur un *plan factoriel* composé de deux axes, et le *cercle des corrélations* (voir fiche **89**) qui permet d'interpréter la répartition des individus.

Graphe des individus

Pour tracer le graphe : `MVA.plot(ACP)`[2]. Les axes 1 (horizontal) et 2 (vertical) sont représentés par défaut, ils peuvent être changés grâce aux arguments `xax` et `yax`.

Pour ajouter des groupes sur le graphe, utiliser l'argument `fac=facteur` où `facteur` est le facteur définissant le groupe de chaque individu. Les arguments `col`, `pch`, `fac.lab`, `contours`, `stars` et `barycenters` permettent de personnaliser une telle représentation.

Voir `?MVA.scoreplot` pour bien d'autres d'options graphiques.

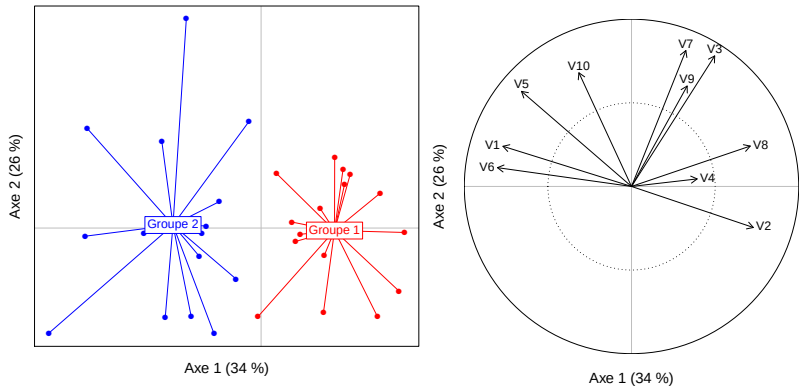
Cercle des corrélations

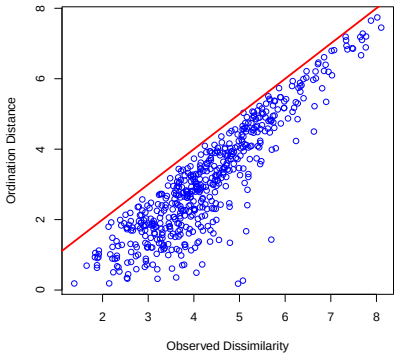
Pour tracer le graphe : `MVA.plot(ACP,"corr")`[2]. Pour supprimer les flèches, ajouter l'argument `arrows=FALSE`. Voir `?MVA.corplot` pour bien d'autres d'options graphiques.

Interprétation

Le graphe des individus permet d'identifier s'il existe une structuration dans le jeu de données (en groupes, le long d'un gradient…). Si des groupes sont suspectés, ils peuvent être déterminés de façon objective grâce à une méthode de *classification* (voir fiches **93** à **96**). On peut également tester si des groupes connus et/ou des covariables « corrèlent » significativement avec la structuration révélée par l'ACP (voir fiche **91**).

Dans un second temps, le cercle des corrélations permet d'identifier les variables du jeu de données qui expliquent la structuration observée. Pour cela, on repère quelles sont les directions pertinentes pour l'interprétation biologique sur le graphe des individus (ce peuvent être des axes ou n'importe quelles diagonales), et on identifie les variables qui corrèlent le plus avec ces directions sur le cercle des corrélations (voir fiche **89**).





98. L'analyse factorielle des correspondances

— PACKAGE(S) ADDITIONNEL(S) UTILISÉ(S) DANS CETTE FICHE —

[1]`vegan`, [2]`RVAideMemoire`

SYNONYMES, TRADUCTIONS ET ABRÉVIATIONS

Français – Analyse factorielle des correspondances (AFC)
Anglais – *Correspondence analysis* (*CA* ou *COA*)

Préparation des données

Le jeu de données est soit un tableau de contingence (*i.e.* valeurs nulles ou entières positives) soit un tableau de présence-absence (*i.e.* 0/1). En AFC les lignes et les colonnes du tableau sont considérées de façon symétrique, *i.e.* il n'y a pas d'« individus » et de « variables ». Lignes et colonnes peuvent donc parfaitement être inversées sans changer l'analyse.

Réalisation de l'analyse

Pour réaliser l'AFC : `AFC<-cca(tableau)`[1] où `tableau` est le tableau de données.

Qualité de l'analyse

L'AFC a pour objectif de synthétiser au mieux une certaine information qui est la *correspondance* entre les lignes et les colonnes du jeu de données, que l'on appellera plus généralement *inertie* (l'inertie est en fait un terme générique, le fait qu'elle représente une correspondance est un cas particulier). Pour estimer la qualité de l'analyse, on s'intéresse donc au pourcentage d'inertie expliqué par chaque axe. Pour obtenir ces pourcentages : `MVA.synt(AFC)`[2].

Remarque 1 : les pourcentages d'inertie sont toujours en ordre décroissant (*i.e.* l'axe 1 explique plus d'inertie que l'axe 2, qui en explique lui-même plus que l'axe 3...).

Remarque 2 : il n'y a pas de règle absolue sur le nombre d'axes à retenir pour l'interprétation. Il s'agit toujours d'un compromis entre une bonne synthèse de l'information du jeu de données (qui augmente avec le nombre d'axes) et une facilité à interpréter (qui diminue avec le nombre d'axes).

Il peut arriver que les pourcentages d'inertie expliqués par l'AFC soient relativement faibles. Cela ne veut pas forcément dire que l'analyse est inutile. En effet, ce qui compte est que les distances interindividuelles dans l'espace multivarié créé par l'analyse soient bien représentatives des distances interindividuelles réelles (*i.e.* dans le tableau de données). Pour vérifier cela, on trace un *diagramme de Shepard* : `stressplot(AFC)`[1]. Sur ce diagramme, si les points sont à peu près alignés le long d'une droite alors les distances dans l'espace de l'AFC sont bien proportionnelles aux distances réelles, et l'on peut se baser sur les résultats de l'analyse pour l'interprétation. Si les points ne dessinent clairement pas une droite, les distances ne sont pas préservées et interpréter l'analyse est inutile car elle ne représente pas la réalité.

Remarque 1 : la droite rouge du diagramme de Shepard indique une proportionnalité parfaite entre les distances. Si les points forment une droite proche de la droite rouge cela est signe d'un pourcentage d'inertie expliqué élevé, s'ils forment une droite plus éloignée que ce pourcentage est moins important (mais tout de même qu'il y a proportionnalité des distances).

Remarque 2 : par défaut le diagramme de Shepard considère les deux premiers axes de l'AFC. Si l'interprétation est basée sur plus d'axes, ajouter l'argument `k=nb` où `nb` est le nombre total d'axes utilisés.

Représentation graphique

En AFC les lignes et les colonnes sont représentées chacune par un point sur le *même graphe*, qu'on pourrait appeler « graphe d'association ». Il est cependant impossible de représenter à la fois les distances interlignes et les distances intercolonnes sans biais sur le même graphe. Il faut donc choisir entre représenter sans biais les premières (échelle de type 1) ou les secondes (échelle de type 2).

Pour représenter les distances interlignes sans biais : `MVA.plot(AFC,points=FALSE,scaling=1)`[2]. Pour représenter les distances intercolonnes sans biais : `MVA.plot(AFC,points=FALSE,scaling=2)`[2]. Les axes 1 (horizontal) et 2 (vertical) sont représentés par défaut, ils peuvent être changés grâce aux arguments `xax` et `yax`. Les arguments `col`, `pch` et `points` permettent de personnaliser une telle représentation. Voir `?MVA.scoreplot` pour bien d'autres d'options graphiques.

Ajouter des groupes sur le graphe est possible mais la procédure est plus complexe :

```
> assoc <- MVA.plot(AFC,points=FALSE,col=couleurs)[2]
```

où `couleurs` est un vecteur à deux valeurs, la première pour les *colonnes* et la seconde pour les *lignes*. S'il l'on souhaite afficher des groupes de colonnes, la première couleur doit être `"white"` ; pour des groupes de lignes la seconde couleur doit être `"white"`.

```
> par(new=TRUE)
> MVA.plot(AFC,points=FALSE,xlim=assoc$xlim,ylim=assoc$ylim,set=nb,fac=facteur)[2]
```

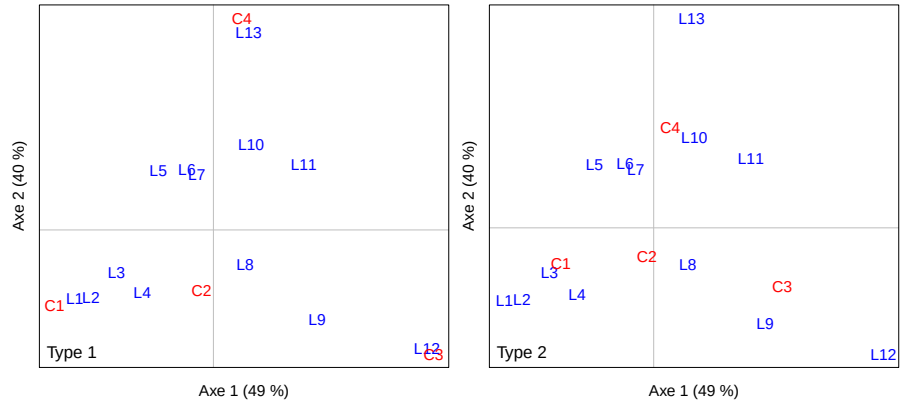
où `nb` vaut 1 pour des groupes de lignes ou 2 pour des groupes de colonnes, et `facteur` est le facteur définissant le groupe de chaque ligne/colonne. Les arguments `col`, `pch`, `fac.lab`, `contours`, `stars` et `barycenters` permettent de personnaliser une telle représentation. Voir `?MVA.scoreplot` pour bien d'autres d'options graphiques.

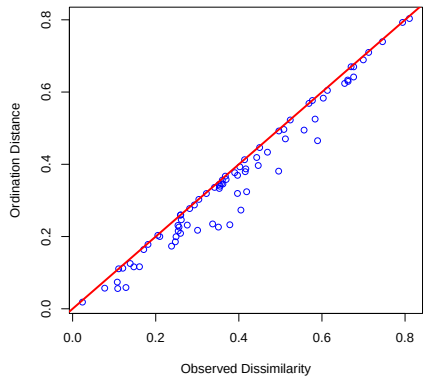
Interprétation

L'idée est relativement simple et repose sur trois principes :

— plus les points représentant deux lignes sont proches et plus ces lignes sont similaires
— idem pour les colonnes
— la proximité entre les points représentant des lignes et des colonnes indique l'association entre ces lignes et ces colonnes.

De manière générale, plus il y a de points éloignés de l'origine du graphe et plus il y a une association forte entre les lignes et les colonnes. Cette association est d'ailleurs directement testable grâce à un test du χ^2 d'indépendance (voir fiche **86**).





99. L'analyse des correspondances multiples

— PACKAGE(S) ADDITIONNEL(S) UTILISÉ(S) DANS CETTE FICHE —

[1]`ade4`, [2]`RVAideMemoire`

SYNONYMES, TRADUCTIONS ET ABRÉVIATIONS

Français – Analyse des correspondances multiples (ACM)
Anglais – *Multiple correspondence analysis* (*MCA*)

Préparation des données

L'ACM est sensible aux effectifs faibles (*i.e.* aux modalités représentées par peu d'individus). Quand ceux-ci sont trop nombreux, mieux vaut regrouper les modalités peu représentées pour obtenir des effectifs plus élevés.

Réalisation de l'analyse

Pour réaliser l'ACM : `ACM<-dudi.acm(tableau,scannf=FALSE,nf=10)`[1] où `tableau` est le tableau de données.

Qualité de l'analyse

L'ACM a pour objectif de synthétiser au mieux une certaine information du jeu de données qu'on appellera de façon générique *inertie*. Pour estimer la qualité de l'analyse, on s'intéresse donc au pourcentage d'inertie expliqué par chaque axe. Pour obtenir ces pourcentages : `MVA.synt(ACM)`[2].

Remarque 1 : les pourcentages d'inertie sont toujours en ordre décroissant (*i.e.* l'axe 1 explique plus d'inertie que l'axe 2, qui en explique lui-même plus que l'axe 3...).

Remarque 2 : il n'y a pas de règle absolue sur le nombre d'axes à retenir pour l'interprétation. Il s'agit toujours d'un compromis entre une bonne synthèse de l'information du jeu de données (qui augmente avec le nombre d'axes) et une facilité à interpréter (qui diminue avec le nombre d'axes).

Représentation graphique

L'unique représentation graphique est le *graphe des individus*, qui montre la position de ces individus sur un *plan factoriel* composé de deux axes. Pour tracer le graphe : `MVA.plot(ACM)`[2]. Les axes 1 (horizontal) et 2 (vertical) sont représentés par défaut, ils peuvent être changés grâce aux arguments `xax` et `yax`.

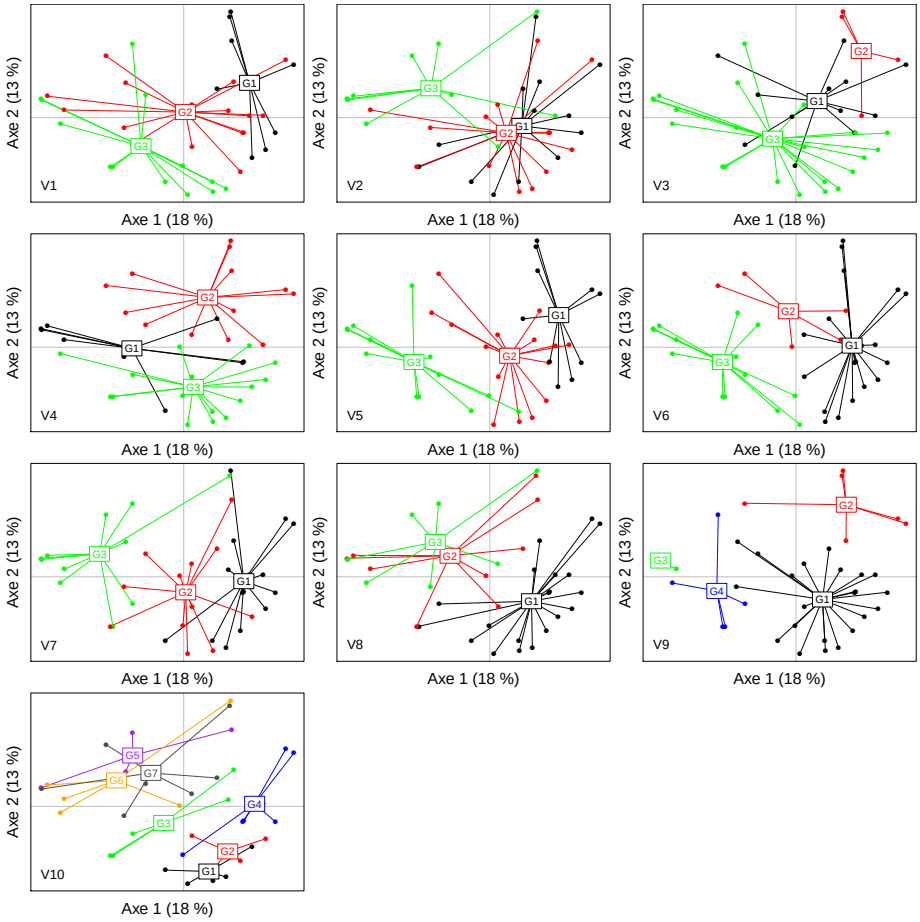
Par défaut le graphe des individus est tracé autant de fois qu'il y a de variables dans le jeu de données, et sur chacun des graphes les modalités d'une seule variable sont représentées. Pour ne représenter qu'une seule variable, ajouter les arguments `byfac=FALSE` et `fac=variable` où `variable` est la variable d'intérêt (nécessairement un facteur). Qu'il y ait un seul ou plusieurs graphes, les arguments `col`, `pch`, `contours`, `stars` et `barycenters` permettent de personnaliser la représentation.

Voir `?MVA.scoreplot` pour bien d'autres d'options graphiques.

Interprétation

L'idée est relativement simple : plus deux modalités de variables différentes sont proches, plus elles sont associées dans le jeu de données. L'interprétation nécessite de superposer mentalement les différents graphes, ce qui complique sérieusement les choses quand le nombre de variables augmente. Mieux vaut donc limiter ce nombre de variables.

Si l'on souhaite interpréter biologiquement un axe de l'analyse (ce qui est intéressant si cet axe est une direction structurante du nuage de points), il faut se concentrer sur les variables dont le poids est le plus important dans la construction de cet axe. En ACM cela se mesure par un *rapport de corrélation*, qui varie entre 0 (poids nul) et 1 (poids très important). Pour obtenir les rapports : `scat.cr(ACM,axis=nb)`[2] où `nb` est le numéro de l'axe choisi.


Axe 2 (13 %)
Axe 1 (18 %)
V1
V2
V3
V4
V5
V6
V7
V8
V9
V10
G1
G2
G3
G4
G5
G6
G7

100. L'analyse mixte

— PACKAGE(S) ADDITIONNEL(S) UTILISÉ(S) DANS CETTE FICHE —

[1]`ade4`, [2]`RVAideMemoire`

SYNONYMES, TRADUCTIONS ET ABRÉVIATIONS

Français – Analyse mixte
Anglais – *Mix analysis*

Préparation des données

Comme l'ACP (voir fiche **97**), l'analyse mixte fonctionne d'autant mieux que les variables quantitatives du jeu de données ont une distribution au moins à peu près symétrique. Une transformation préalable de ces variables peut grandement aider à améliorer la situation (voir fiche **88**).

Comme l'ACM (voir fiche **99**), l'analyse mixte est sensible aux effectifs faibles (*i.e.* aux modalités des variables qualitatives représentées par peu d'individus). Quand ceux-ci sont trop nombreux, mieux vaut regrouper les modalités peu représentées pour obtenir des effectifs plus élevés.

Réalisation de l'analyse

Pour réaliser l'analyse mixte : `AMix<-dudi.mix(tableau,scannf=FALSE,nf=10)`[1] où `tableau` est le tableau de données. Les variables quantitatives sont standardisées (voir fiche **88**) automatiquement.

Qualité de l'analyse

L'analyse mixte a pour objectif de synthétiser au mieux une certaine information du jeu de données qu'on appellera de façon générique *inertie*. Pour estimer la qualité de l'analyse, on s'intéresse donc au pourcentage d'inertie expliqué par chaque axe. Pour obtenir ces pourcentages : `MVA.synt(AMix)`[2].

Remarque 1 : les pourcentages d'inertie sont toujours en ordre décroissant (*i.e.* l'axe 1 explique plus d'inertie que l'axe 2, qui en explique lui-même plus que l'axe 3...).

Remarque 2 : il n'y a pas de règle absolue sur le nombre d'axes à retenir pour l'interprétation. Il s'agit toujours d'un compromis entre une bonne synthèse de l'information du jeu de données (qui augmente avec le nombre d'axes) et une facilité à interpréter (qui diminue avec le nombre d'axes).

Représentations graphiques

À partir du moment où il y a au moins une variable quantitative dans le jeu de données, deux représentations sont possibles : le *graphe des individus* qui montre la position de ces individus sur un *plan factoriel* composé de deux axes, et le *cercle des corrélations* (voir fiche **89**). S'il n'y a aucune variable quantitative le seul graphe possible est celui des individus.

Graphe des individus

Pour tracer le graphe : `MVA.plot(AMix)`[2]. Les axes 1 (horizontal) et 2 (vertical) sont représentés par défaut, ils peuvent être changés grâce aux arguments `xax` et `yax`.

Par défaut le graphe des individus est tracé autant de fois qu'il y a de variables qualitatives dans le jeu de données, et sur chacun des graphes les modalités d'une seule variable sont représentées. Pour ne représenter qu'une seule variable, ajouter les arguments `byfac=FALSE` et `fac=variable` où `variable` est la variable d'intérêt (nécessairement un facteur). Qu'il y ait un seul ou plusieurs graphes, les arguments `col`, `pch`, `contours`, `stars` et `barycenters` permettent de personnaliser la représentation.

Voir `?MVA.scoreplot` pour bien d'autres d'options graphiques.

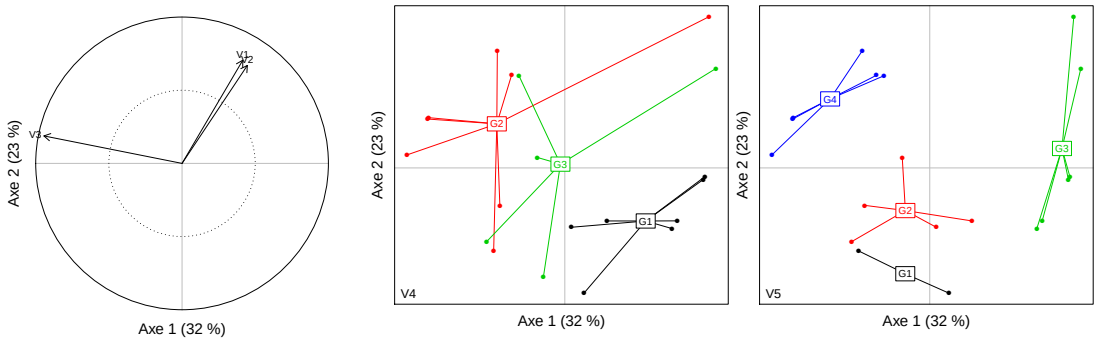
Cercle des corrélations

Pour tracer le graphe : `MVA.plot(AMix,"corr")`[2]. Pour supprimer les fèches, ajouter l'argument `arrows=FALSE`. Voir `?MVA.corplot` pour bien d'autres d'options graphiques.

Interprétation

L'analyse mixte est une sorte d'intermédiaire entre l'ACP (à laquelle est est équivalente s'il n'y a que des variables quantitatives) et l'ACM (à laquelle est équivalente s'il n'y a que des variables qualitatives nominales, *i.e.* non ordonnées). L'interprétation se fait donc comme en ACP pour les variables quantitatives (voir fiche **97**), et comme en ACM pour les variables qualitatives (voir fiche **99**). Les relations entre variables quantitatives et qualitatives s'interprètent simplement : plus le barycentre d'une modalité est situé vers l'avant d'une flèche (quand on le projette perpendiculairement à cette flèche) et plus la valeur moyenne des individus de cette modalité est élevée pour la variable quantitative représentée par la fèche (et vice-versa, voir fiche **89**).

Si l'on souhaite interpréter biologiquement un axe de l'analyse (ce qui est intéressant si cet axe est une direction structurante du nuage de points), il faut se concentrer sur les variables dont le poids est le plus important dans la construction de cet axe. En analyse mixte cela se mesure par différents indicateurs selon la nature des variables, mais qui varient tous entre 0 (poids nul) et 1 (poids très important). Pour obtenir ces indicateurs : `scat.cr(AMix,axis=nb)`[2] où `nb` est le numéro de l'axe choisi.



101. Les matrices de distance

— PACKAGE(S) ADDITIONNEL(S) UTILISÉ(S) DANS CETTE FICHE —

[1]ade4, [2]vegan, [3]factoextra, [4]Hotelling, [5]cluster

Une matrice de distance est une matrice dans laquelle chaque paire d'individus est caractérisée par une distance les séparant. Ce sont donc des matrices (i) carrées (*i.e.* autant de lignes et de colonnes que d'individus), (ii) symétriques (car la distance entre A et B est la même qu'entre B et A) et (iii) dont la diagonale – qui sépare deux triangles symétriques, l'un inférieur et l'autre supérieur – est constituée de 0. Par souci d'affichage et de stockage, seul le triangle inférieur est souvent créé.

— EXEMPLE(S) —

La même matrice de distance (entre trois individus ici) peut être représentée entièrement ou par l'un de ses triangles (habituellement l'inférieur), qui contient à lui seul toute l'information :

```
  A    B    C
A 0.00 2.17 2.10
B 2.17 0.00 0.29
C 2.10 0.29 0.00
```

```
  A    B
B 2.17
C 2.10 0.29
```

Pour certains types de données on travaille nécessairement sur des matrices de distance (distances géographiques, distances basées sur des séquences (de nucléotides ou d'acides aminés par exemple)...). Pour d'autres, travailler sur une matrice de distance permet de synthétiser l'information contenue dans plusieurs variables. Dans ce dernier cas cependant, la contribution de chaque variable aux distances interindividuelles est perdue. Quand le jeu de données est un tableau de variables, on a donc le choix entre deux types d'analyses : sur les variables ou sur une matrice de distance. Il est relativement simple de décider : si la question porte (et donc si l'interprétation biologique repose) sur les variables, utiliser une analyse sur les variables. Si la question porte sur la (dis)similarité globale entre individus, utiliser une analyse sur une matrice de distance.

Distance et similarité sont des mesures totalement liées (mais opposées), *i.e.* plus la distance (ou dissimilarité) entre deux individus est élevée, plus leur similarité est faible. De nombreuses distances sont ainsi calculées à partir d'un indice de similarité, par la simple relation : $Distance = 1 - Similarité$.

Une matrice de distance est dite *euclidienne* si elle est représentable dans un espace multidimensionnel euclidien (*i.e.* l'espace géométrique traditionnel). Il est important de savoir si une matrice de distance est euclidienne ou non, car un certain nombre de méthodes d'analyse exigent une matrice euclidienne. Quelle que soit la matrice de distance, on peut tester simplement si elle est euclidienne *via* `is.euclid(mat.dist)`[1] où `mat.dist` est la matrice. Pour rendre une matrice de distance euclidienne, une solution simple et quasiment toujours efficace est d'utiliser la racine carrée : `mat.dist2<-sqrt(mat.dist)`.

Remarque : en pratique, beaucoup de distances usuelles sont rendues euclidiennes en étant calculées de cette manière : $Distance = \sqrt{1 - Similarité}$.

Le choix d'une mesure de distance dépend avant tout du type de données dont on dispose, mais aussi de son champ disciplinaire. Il n'est donc pas question ici de toutes les comparer. Les fonctions permettant de calculer les distances les plus courantes sont présentées, selon le type de données.

Données binaires (0/1)

Dix distances – basées sur des indices de similarité – sont proposées par la fonction `dist.binary()`[1], dont les plus classiques (indices de Jaccard, de Sokal & Michener ou de Sørensen - Dice). Elles sont numérotées de 1 à 10 (voir `?dist.binary` pour la liste complète). Pour calculer la matrice de distance :

`dist.binary(tableau,method=numero)`[1] où `tableau` est le tableau de données et `numero` le numéro de la distance choisie. Toutes les matrices renvoyées par cette fonction sont euclidiennes.

Le point important avec les données binaires est la question du *double zéro, i.e.* de l'absence simultanée de ce que l'on mesure chez deux individus. Cette absence peut en effet avoir du sens, ou non. Les indices qui prennent en compte la double absence comme preuve d'une similarité (comme celui de Sokal & Michener) sont dits *symétriques*, tandis que ceux qui font le postulat inverse (comme ceux de Jaccard et de Sørensen - Dice) sont dits *asymétriques*.

Tableaux de contingence (*i.e.* croisement des modalités de deux facteurs) ou tableaux d'abondance

Dans ces deux cas, les valeurs ne peuvent être qu'entières positives ou nulles.

Les distances les plus classiquement employées avec de telles données sont :

— la distance de Bray-Curtis : `vegdist(tableau,method="bray")`[2]. La matrice renvoyée n'est pas euclidienne.

— la distance de Chord : `vegdist(decostand(tableau,method="normalize"),method="euclidean")`[2]. La matrice renvoyée est euclidienne.

— la distance d'Hellinger : `vegdist(decostand(tableau,method="hellinger"),method="euclidean")`[2]. La matrice renvoyée est euclidienne.

— la distance du χ^2 : `vegdist(decostand(tableau,method="chi.square"),method="euclidean")`[2]. La matrice renvoyée est euclidienne.

Données quantitatives (hors données compositionnelles)

La fonction `dist()` permet de calculer les distances principales (Euclidienne, Manhattan...). Pour l'utiliser : `dist(tableau,method="methode")` où `methode` est le nom de la méthode entre guillemets (voir `?dist` pour la liste complète). D'autres mesures de distance sont proposées par `dist.quant()`[1], qui fonctionne avec des numéros de méthode (voir `?dist.quant` pour la liste complète).

Une alternative à calculer les distances à partir des valeurs des variables (ce que font les fonctions précédentes) est d'utiliser la *corrélation* entre les individus. Pour calculer de telles distances : `get_dist(tableau,method="pearson")`[3].

Remarque : pour des données quantitatives il peut être intéressant (et même recommandé si l'on souhaite faire de la classification) de standardiser le tableau de données avant de calculer les distances, afin de donner le même poids à toutes les variables (voir fiche **88**).

Données compositionnelles

Un jeu de données est dit *compositionnel* quand la somme des valeurs d'un individu vaut toujours 1 ou 100 %. On parle aussi de proportions relatives.

Une distance classique avec de telles données est la distance d'Aitchison. Elle s'obtient en trois étapes :

1. Éventuellement, standardiser le tableau de données afin de donner le même poids à toutes les variables (ce qui est recommandé en particulier si l'on souhaite faire de la classification ; voir fiche **88**), grâce à la fonction `scale()`.
2. Appliquer la transformation du *Centered LogRatio* (CLR) au résultat de l'étape précédente, grâce à la fonction `clr()`[4] (voir fiche **88**).
3. Appliquer la distance Euclidienne au résultat de l'étape précédente, grâce à la fonction `dist()`. La matrice de distance finalement obtenue est euclidienne.

EXEMPLE(S)

Le jeu de données `tab` est compositionnel. On souhaite obtenir une matrice de distance basée sur la distance d'Aitchison, et donner le même poids à toutes les variables :

```
> tab.stand <- scale(tab)
> tab.stand.clr <- clr(tab.stand)[4]
> mat.dist <- dist(tab.stand.clr,method="euclidean")
```

Données mixtes (variables de plusieurs types)

La distance utilisée est celle de Gower, qui permet de traiter à la fois les variables quantitatives, binaires (codées 0/1), ordinales et nominales. La matrice est calculée *via* `daisy(tableau)`[5]. Les variables qualitatives nominales (*i.e.* facteurs non ordonnés) sont reconnues comme telles, tout comme les variables qualitatives ordinales (*i.e.* facteurs ordonnés, voir fiche **52** pour les créer). Pour les variables binaires, mieux vaut préciser explicitement si elles doivent être considérées de manière symétrique ou non. Ajouter pour cela l'argument `type=list(symm=var.sym,asymm=var.asym)` où `var.sym` est un vecteur donnant le nom (ou le numéro) des colonnes de `tableau` à considérer symétriquement, et `var.asym` la même chose pour les variables à considérer asymétriquement. La matrice renvoyée n'est pas euclidienne.

Données génétiques

L'analyse de données génétiques est un monde en soi. Beaucoup de méthodes sont proposées par le package `adegenet`, qui travaillent sur des données formatées d'une façon particulière. Voir `?adegenet` (après avoir chargé le package) pour une explication détaillée.

Finalement, il est possible d'importer dans **R** une matrice de distance quelconque, qui sera considérée comme un tableau. Il est nécessaire de transformer ce tableau en un objet de type matrice de distance pour qu'il puisse être utilisé comme tel. Pour cela : `mat.dist<-as.dist(tableau)`.

102. L'analyse en coordonnées principales

— PACKAGE(S) ADDITIONNEL(S) UTILISÉ(S) DANS CETTE FICHE —

[1]`vegan`, [2]`RVAideMemoire`

SYNONYMES, TRADUCTIONS ET ABRÉVIATIONS

Français – Analyse en coordonnées principales
Positionnement multidimensionnel métrique
Positionnement multidimensionnel classique

Anglais – *Principal coordinate analysis* (*PCoA* ou *PCO*)
Metric multidimensional scaling (*MDS*)
Classical multidimensional scaling (*MDS*)
Classical scaling

Préparation des données

Il est indispensable de vérifier si la matrice de distance est euclidienne ou non (voir fiche **101**).

Réalisation de l'analyse

Pour réaliser la PCoA : `PCoA<-dbrda(mat.dist~1)`[1] où `mat.dist` est la matrice de distance. Si celle-ci n'est pas euclidienne, il est nécessaire d'appliquer une correction. Ajouter pour cela l'argument `add=TRUE`.

Qualité de l'analyse

La PCoA a pour objectif de synthétiser au mieux une certaine information qui est la *variance totale* de la matrice de distance. Pour estimer la qualité de l'analyse, on s'intéresse donc au pourcentage de variance expliqué par chaque axe. Pour obtenir ces pourcentages : `MVA.synt(PCoA)`[2].

Remarque 1 : les pourcentages de variance sont toujours en ordre décroissant (*i.e.* l'axe 1 explique plus de variance que l'axe 2, qui en explique lui-même plus que l'axe 3...).

Remarque 2 : il n'y a pas de règle absolue sur le nombre d'axes à retenir pour l'interprétation. Il s'agit toujours d'un compromis entre une bonne synthèse de l'information du jeu de données (qui augmente avec le nombre d'axes) et une facilité à interpréter (qui diminue avec le nombre d'axes).

Il peut arriver que les pourcentages de variance expliqués par la PCoA soient relativement faibles. Cela ne veut pas forcément dire que l'analyse est inutile. En effet, ce qui compte est que les distances interindividuelles dans l'espace multivarié créé par l'analyse soient bien représentatives des distances interindividuelles réelles. Pour vérifier cela, on trace un *diagramme de Shepard* : `stressplot(PCoA)`[1]. Sur ce diagramme, si les points sont à peu près alignés le long d'une droite alors les distances dans l'espace de la PCoA sont bien proportionnelles aux distances réelles, et l'on peut se baser sur les résultats de l'analyse pour l'interprétation. Si les points ne dessinent clairement pas une droite, les distances ne sont pas préservées et interpréter l'analyse est inutile car elle ne représente pas la réalité.

Remarque 1 : la droite rouge du diagramme de Shepard indique une proportionnalité parfaite entre les distances. Si les points forment une droite proche de la droite rouge cela est signe d'un pourcentage de variance expliqué élevé, s'ils forment une droite plus éloignée que ce pourcentage est moins important (mais tout de même qu'il y a proportionnalité des distances).

Remarque 2 : par défaut le diagramme de Shepard considère les deux premiers axes de la PCoA. Si l'interprétation est basée sur plus d'axes, ajouter l'argument `k=nb` où `nb` est le nombre total d'axes utilisés.

Représentation graphique

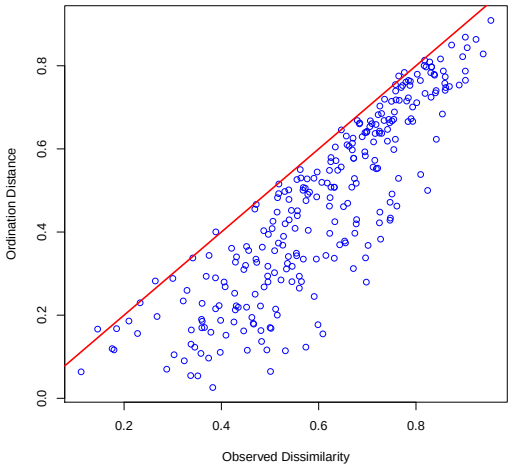
L'unique représentation graphique est le *graphe des individus*, qui montre la position de ces individus sur un *plan factoriel* composé de deux axes. Pour tracer le graphe : `MVA.plot(PCoA)`[2]. Les axes 1 (horizontal) et 2 (vertical) sont représentés par défaut, ils peuvent être changés grâce aux arguments `xax` et `yax`.

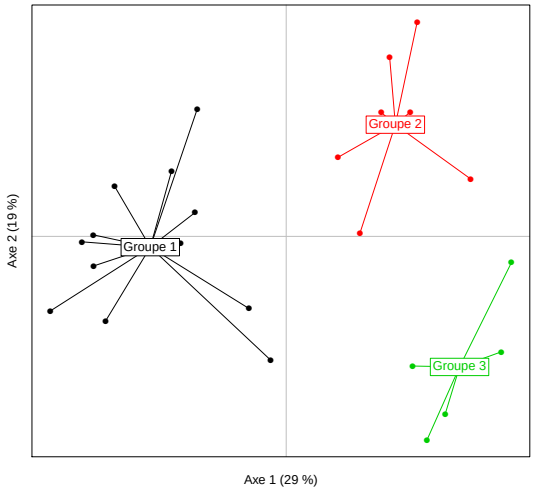
Pour ajouter des groupes sur le graphe, utiliser l'argument `fac=facteur` où `facteur` est le facteur définissant le groupe de chaque individu. Les arguments `col`, `pch`, `fac.lab`, `contours`, `stars` et `barycenters` permettent de personnaliser une telle représentation.

Voir `?MVA.scoreplot` pour bien d'autres d'options graphiques.

Interprétation

Le graphe des individus permet d'identifier s'il existe une structuration dans le jeu de données (en groupes, le long d'un gradient...). Si des groupes sont suspectés, ils peuvent être déterminés de façon objective grâce à une méthode de *classification* (voir fiches **93** à **96**). On peut également tester si des groupes connus et/ou des covariables « corrèlent » significativement avec la structuration révélée par la PCoA (voir fiche **91**).





103. Le positionnement multidimensionnel non métrique

— PACKAGE(S) ADDITIONNEL(S) UTILISÉ(S) DANS CETTE FICHE —

[1]vegan, [2]RVAideMemoire

SYNONYMES, TRADUCTIONS ET ABRÉVIATIONS

Français – Positionnement multidimensionnel non métrique
Anglais – *Non metric multidimensional scaling* (*nMDS*)

Réalisation de l'analyse

Pour réaliser la nMDS : `nMDS<-metaMDS(mat.dist,k=nb.axes,trace=FALSE)`[1] où `mat.dist` est la matrice de distance et `nb.axes` le nombre d'axes à construire. L'argument `trace=FALSE` permet de ne pas afficher les étapes intermédiaires du calcul.

Remarque : la nMDS est une analyse où la construction des axes dépend du nombre d'axes à créer. Son résultat est donc différent selon ce nombre.

Qualité de l'analyse

La qualité d'une nMDS s'évalue grâce à un indicateur nommé *stress*, qui varie entre 0 et 1 (parfois entre 0 et 100 %). Cette valeur est obtenue *via* `MVA.synt(nMDS)`[2]. De façon empirique on considère qu'un stress < 0.05 est synonyme d'une excellente synthèse, < 0.1 est bon, < 0.2 est correct et > 0.2 est suspect (voire mauvais).

Une autre manière d'évaluer la qualité d'une nMDS est de tester si les distances interindividuelles dans l'espace multivarié créé par l'analyse sont bien représentatives des distances interindividuelles réelles (ou plutôt de l'*ordre* de ces distances puisque la nMDS n'utilise pas les distances réelles mais seulement une information semi-quantitative : « telle distance est plus grande que telle autre »). On trace pour cela un *diagramme de Shepard* : `stressplot(nMDS)`[1]. Sur ce diagramme, si les points restent proches de la courbe en escalier rouge l'ordre des distances est bien préservé et l'on peut se baser sur les résultats de l'analyse pour l'interprétation. Si les points ne se regroupent clairement pas autour de la courbe rouge, l'ordre des distances n'est pas préservé et interpréter l'analyse est inutile car elle ne représente pas la réalité.

Remarque : le stress est en fait un indicateur global du regroupement des points autour de la courbe du diagramme de Shepard.

Représentation graphique

L'unique représentation graphique est le *graphe des individus*, qui montre la position de ces individus sur un *plan factoriel* composé de deux axes. Pour tracer le graphe : `MVA.plot(nMDS)`[2]. Les axes 1 (horizontal) et 2 (vertical) sont représentés par défaut, ils peuvent être changés grâce aux arguments `xax` et `yax` (si d'autres axes ont été construits).

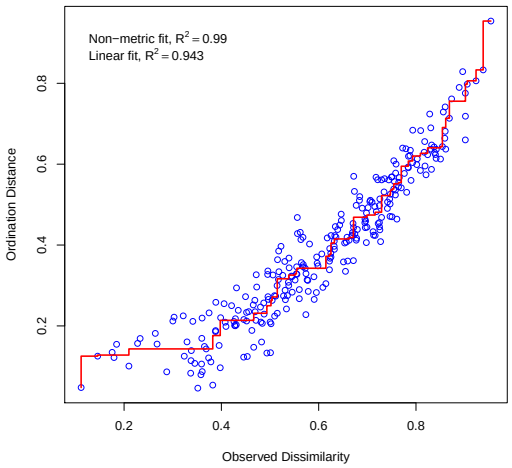
Pour ajouter des groupes sur le graphe, utiliser l'argument `fac=facteur` où `facteur` est le facteur définissant le groupe de chaque individu. Les arguments `col`, `pch`, `fac.lab`, `contours`, `stars` et `barycenters` permettent de personnaliser une telle représentation.

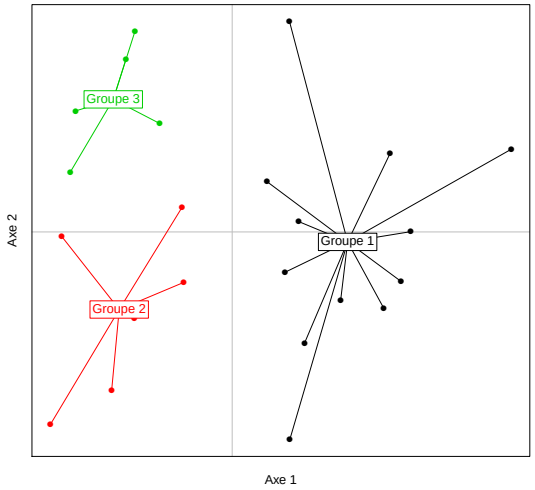
Voir `?MVA.scoreplot` pour bien d'autres d'options graphiques.

Interprétation

Le graphe des individus permet d'identifier s'il existe une structuration dans le jeu de données (en groupes, le long d'un gradient…). Si des groupes sont suspectés, ils peuvent être déterminés de façon objective grâce à une méthode de *classification* (voir fiches **93** à **96**). Attention cependant, la nMDS ne préserve pas les distances réelles entre individus, mais seulement une information relative. Ce n'est donc pas parce qu'un point est deux fois plus éloigné d'un autre que d'un troisième que la distance réelle entre

le premier et le troisième est deux fois plus grande que la distance entre le premier et le deuxième. Il n'y a ainsi pas vraiment de sens à « corréler » les résultats d'une nMDS à des variables externes.





104. L'analyse de redondance

— PACKAGE(S) ADDITIONNEL(S) UTILISÉ(S) DANS CETTE FICHE —

[1]`vegan`, [2]`RVAideMemoire`

SYNONYMES, TRADUCTIONS ET ABRÉVIATIONS

Français – Analyse de redondance
Analyse en composantes principales sur variables instrumentales (ACPVI)

Anglais – *Redundancy analysis* (*RDA*)
Principal component analysis with respect to instrumental variables (*PCAIV*)
Principal components of instrumental variables

Préparation des données

La RDA fonctionne d'autant mieux que les variables à expliquer ont une distribution à peu près normale (au moins symétrique), qu'elles sont reliées entre elles par des relations linéaires et qu'elles sont reliées aux variables explicatives par des relations linéaires. Une transformation préalable du tableau à expliquer peut grandement aider à améliorer la situation (voir fiche **88**).

Il est également nécessaire que les matrices de variance-covariance (l'équivalent multivarié de la variance) soient homogènes entre les différentes modalités des variables explicatives qualitatives (s'il y en a). Pour le tester : `anova(betadisper(dist(tableau),facteur))`[1] où `tableau` est le tableau *à expliquer* et `facteur` le facteur définissant les groupes.

Enfin, il est recommandé la plupart du temps de standardiser les variables à expliquer avant l'analyse (voir fiche **88**). Cela permet de donner le même poids à toutes les variables, et d'interpréter les résultats en termes de corrélation ce qui est souvent plus facile. Dans cette fiche on considèrera que les variables sont standardisées.

Réalisation de l'analyse

Pour réaliser la RDA : `RDA<-rda(formule,data=tab.explicatif)`[1] où `tab.explicatif` est le tableau contenant les variables explicatives. Voir fiche **40** pour une explication détaillée de la construction d'une formule. Dans cette formule, la réponse est `tableau` (*i.e.* le tableau à expliquer). Si l'on souhaite standardiser les variables de ce tableau mais que l'on n'a pas effectué l'opération au préalable, ajouter l'argument `scale=TRUE`. Par défaut les variables *ne sont pas standardisées*.

Capacité explicative globale

La RDA consiste en fait en deux étapes :

1. Séparer la variation (du tableau à expliquer) due aux variables explicatives (appelée variation *contrainte*) de la variation non expliquée (dite *résiduelle* ou *non contrainte*). La RDA travaille sur une certaine variation qui est la *variance*.
2. Réaliser deux ACP séparées, l'une sur la variation contrainte (« ACP contrainte ») et l'autre sur la variation non contrainte (« ACP non contrainte »).

On peut estimer la capacité explicative globale de la RDA grâce au pourcentage de variance contrainte de l'analyse (*i.e.* de variance du tableau à expliquer expliquée par les variables explicatives). Plus ce pourcentage est élevé et plus la variation observée dans le tableau à expliquer est liée aux variables explicatives. Ce pourcentage est obtenu *via* `MVA.synt(RDA)`[2], dans le premier tableau renvoyé par la fonction.

Qualité de l'analyse

Test(s)

L'effet des variables explicatives est testé *via* `MVA.anova(RDA)`[2]. Un test F par permutation est réalisé.

Si au moins une variable explicative a un effet significatif, on peut se baser sur les résultats de l'ACP *contrainte* pour l'interprétation. Si aucune variable explicative n'a d'effet significatif, interpréter les résultats de cette ACP n'a pas beaucoup d'intérêt puisqu'aucun effet n'est montré.

En lien avec cette ACP contrainte, on peut réaliser des comparaisons multiples entre modalités d'un facteur (ou combinaisons de modalités d'une interaction entre facteurs) à effet significatif. Pour réaliser le test : `pairwise.factorfit(RDA,facteur)`[2] où `facteur` est le facteur d'intérêt.

Synthèse

Si au moins une variable explicative a un effet significatif, on s'intéresse à l'ACP contrainte. Comme pour une ACP classique (voir fiche **97**), on estime la qualité de cette analyse par le pourcentage de variance expliqué par chaque axe. Ces pourcentages sont obtenus *via* `MVA.synt(RDA)`[2], dans le deuxième tableau renvoyé par la fonction.

Remarque 1 : il s'agit ici de pourcentages de variance *contrainte*, pas totale comme en ACP classique.

Remarque 2 : les pourcentages de variance sont toujours en ordre décroissant (*i.e.* l'axe 1 explique plus de variance que l'axe 2, qui en explique lui-même plus que l'axe 3...).

Remarque 3 : il n'y a pas de règle absolue sur le nombre d'axes à retenir pour l'interprétation. Il s'agit toujours d'un compromis entre une bonne synthèse de l'information (qui augmente avec le nombre d'axes) et une facilité à interpréter (qui diminue avec le nombre d'axes).

Représentations graphiques

En RDA on a deux représentations possibles : le *graphe des individus* qui montre la position de ces individus sur un *plan factoriel* composé de deux axes, et le *cercle des corrélations* (voir fiche **89**) qui permet d'interpréter la répartition des individus.

Graphe des individus

Pour tracer le graphe : `MVA.plot(RDA)`[2]. Les axes 1 (horizontal) et 2 (vertical) sont représentés par défaut, ils peuvent être changés grâce aux arguments `xax` et `yax`. Par défaut c'est l'ACP contrainte qui est représentée. Pour représenter l'ACP non contrainte, ajouter l'argument `space=2`.

Pour ajouter des groupes sur le graphe, utiliser l'argument `fac=facteur` où `facteur` est le facteur définissant le groupe de chaque individu. Les arguments `col`, `pch`, `fac.lab`, `contours`, `stars` et `barycenters` permettent de personnaliser une telle représentation.

Voir `?MVA.scoreplot` pour bien d'autres d'options graphiques.

Cercle des corrélations

Pour tracer le graphe : `MVA.plot(RDA,"corr")`[2]. Comme pour le graphe des individus c'est l'ACP contrainte qui est représentée, l'argument `space=2` permettant de représenter l'ACP non contrainte. Par défaut à la fois les variables à expliquer et les variables explicatives quantitatives sont représentées. Pour ne représenter que les variables explicatives quantitatives ajouter l'argument `set=1`, pour les variables à expliquer `set=2`.

Pour supprimer les flèches, ajouter l'argument `arrows=FALSE`. Voir `?MVA.corplot` pour bien d'autres d'options graphiques.

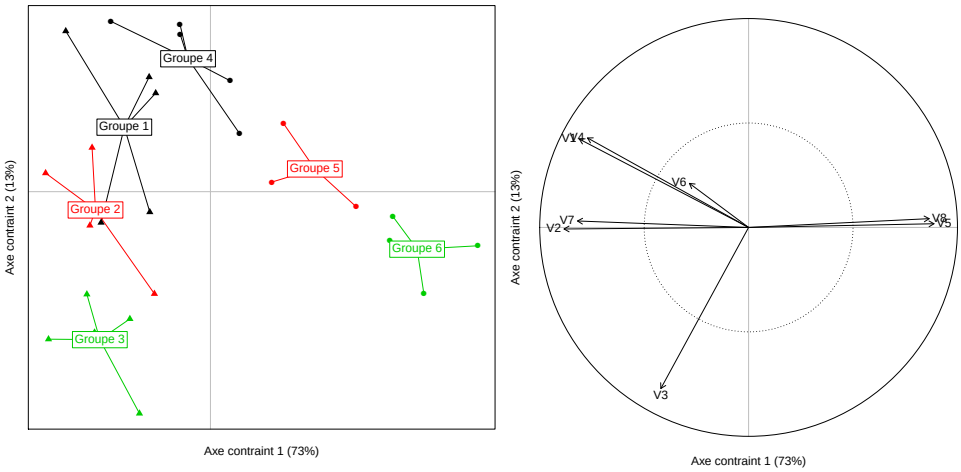
Interprétation

On ne considère que l'ACP contrainte puisque par définition c'est la seule qui permette d'interpréter les résultats en lien avec les variables explicatives.

Le graphe des individus permet d'identifier la structuration des données du tableau à expliquer qui est due aux variables explicatives. On y repère comment les modalités d'un facteur à effet significatif se répartissent, ou des gradients linéaires.

Dans un second temps, le cercle des corrélations permet d'identifier les variables (i) qui différencient d'éventuels groupes et/ou (ii) qui expliquent d'éventuels gradients. Pour cela, on repère quelles sont les directions pertinentes pour l'interprétation biologique sur le graphe des individus (ce peuvent être des axes ou n'importe quelles diagonales), et on identifie les variables qui corrèlent le plus avec ces directions

sur le cercle des corrélations (voir fiche **89**). La façon de procéder est la même pour les variables à expliquer que pour les variables explicatives quantitatives, sauf bien sûr sur que les variables à expliquer sont influencées par les variables explicatives mais pas l'inverse.



105. L'analyse discriminante linéaire

PACKAGE(S) ADDITIONNEL(S) UTILISÉ(S) DANS CETTE FICHE

[1]`vegan`, [2]`MASS`, [3]`RVAideMemoire`

SYNONYMES, TRADUCTIONS ET ABRÉVIATIONS

Français – Analyse discriminante linéaire (ADL)
Anglais – *Linear discriminant analysis* (*LDA*)
Canonical variate analysis (*CVA*)
Discriminant function analysis (*DFA*)

Préparation des données

Les variables quantitatives doivent avoir une distribution normale multivariée. Voir fiche **65** pour tester cette condition. La LDA est toutefois assez robuste si elle n'est pas tout à fait respectée.

Il est également nécessaire que les matrices de variance-covariance (l'équivalent multivarié de la variance) soient homogènes entre les différents groupes. Pour le tester : `anova(betadisper(dist(tableau),facteur)`[1]`)` où `tableau` est le tableau de variables quantitatives et `facteur` le facteur définissant les groupes. Là encore la LDA est assez robuste à un non respect (modéré) de cette condition.

Si ces deux conditions ne sont pas du tout respectées, une transformation préalable des données peut grandement aider à améliorer la situation (voir fiche **88**).

Remarque : les variables quantitatives doivent être standardisées (voir fiche **88**), mais en pratique les fonctions réalisant la LDA le feront automatiquement

Réalisation de l'analyse

Pour réaliser la LDA : `LDA<-lda(tableau,facteur)`[2].

Qualité de l'analyse

Pourcentage d'erreur de classification

Une des fonctions de toute analyse discriminante est de faire de la *prédiction, i.e.* de prédire le groupe auquel appartient un individu dont on connaît seulement la valeur pour les variables quantitatives. Une manière d'estimer la qualité d'une analyse discriminante est donc de tester à quel point elle permet de classer un individu de groupe inconnu sans erreur. On utilise pour cela une méthode de *validation croisée*, qui va générer plusieurs sous-modèles chacun sur une partie du jeu de données (définie aléatoirement) et prédire le groupe des individus non pris en compte dans le modèle. Pour plus de fiabilité l'ensemble de la procédure peut être répété plusieurs fois (avec à chaque fois un découpage aléatoire du jeu de données).

Pour réaliser la validation croisée : `MVA.cv(tableau,facteur,model="LDA")`[3]. Par défaut la fonction découpe le jeu de données en 7 parties (*7-fold cross-validation*), mais ce chiffre peut être modifié *via* l'argument `k=nb` où `nb` est le nombre de sous-jeux de données à générer. La fonction peut également ajuster ce chiffre automatiquement si au moins un groupe contient moins de 7 individus. Par défaut l'ensemble de la procédure est répété 10 fois (argument `repet`), ce qui au final génère $7 \times 10 = 70$ sous-modèles.

Si l'on souhaite utiliser la LDA dans un but prédictif, stocker le résultat de la fonction `MVA.cv()`[3] dans un objet.

Test(s)

Le test à réaliser est une MANOVA (*Multivariate ANalysis Of VAriance*), une extenstion de l'ANOVA au cas multivarié. Les conditions d'emploi de ce test sont les mêmes que celles de la LDA. Voir fiche **109** pour réaliser le test.

Si le facteur a un effet significatif et qu'il y a plus de deux groupes, il est nécessaire de réaliser des comparaisons multiples pour identifier les modalités qui diffèrent. Voir fiche **43** pour réaliser ces comparaisons.

Remarque 1 : pourcentage d'erreur de classification et significativité du facteur sont toujours cohérents : si le pourcentage d'erreur est faible, le facteur a un effet significatif (et inversement).

Remarque 2 : si le facteur n'a pas d'effet significatif, l'interprétation des résultats de la LDA n'a pas vraiment de sens et toute tentative de prédiction est inutile.

Représentations graphiques

En LDA on a deux représentations possibles : le *graphe des individus* qui montre la position de ces individus sur un *plan factoriel* composé de deux axes, et le *cercle des corrélations* (voir fiche **89**) qui permet d'interpréter la répartition des individus.

Remarque : s'il n'y a que deux groupes, la LDA ne produit qu'un seul axe. Le graphe des individus et le cercle des corrélations se réduisent donc à une seule dimension.

Graphe des individus

Pour tracer le graphe : `MVA.plot(LDA,fac=facteur)`[2]. Les axes 1 (horizontal) et 2 (vertical) sont représentés par défaut (s'il y a au moins deux axes), ils peuvent être changés grâce aux arguments `xax` et `yax`.

Dans le cas d'un graphe à deux dimensions, les arguments `col`, `pch`, `fac.lab`, `contours`, `stars` et `barycenters` permettent de personnaliser la représentation. Dans le cas d'un graphe à une seule dimension, les arguments `col`, `legend` et `legend.lab` sont intéressants. Voir `?MVA.scoreplot` pour bien d'autres d'options graphiques.

Cercle des corrélations

Pour tracer le graphe : `MVA.plot(LDA,"corr")`[2].

Pour supprimer les flèches, ajouter l'argument `arrows=FALSE`. Voir `?MVA.corplot` pour bien d'autres d'options graphiques.

Interprétation

Le graphe des individus permet d'identifier comment les groupes se structurent, *i.e.* si tout ou partie des groupes se séparent ou au contraire se chevauchent.

Dans un second temps, le cercle des corrélations permet d'identifier les variables qui différencient ces groupes. Pour cela, on repère quelles sont les directions pertinentes pour l'interprétation biologique sur le graphe des individus (ce peuvent être des axes ou n'importe quelles diagonales), et on identifie les variables qui corrèlent le plus avec ces directions sur le cercle des corrélations (voir fiche **89**).

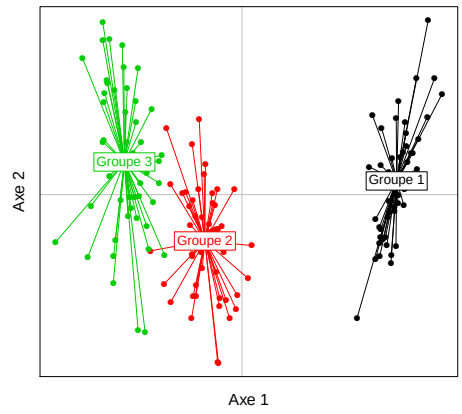
Prédiction

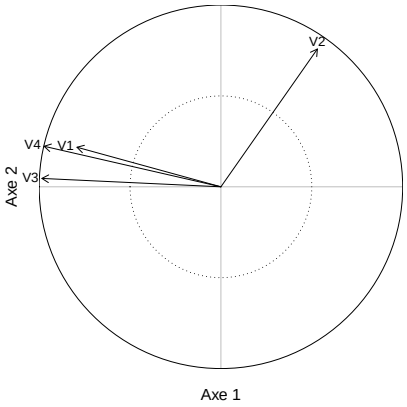
L'un des intérêts des analyses discriminantes est à la fois de comprendre comment les groupes se différencient, mais également de pouvoir prédire le groupe d'un individu pour des valeurs *connues* des variables quantitatives. Prédire un groupe nécessite donc de fixer la valeur de *toutes* les variables quantitatives.

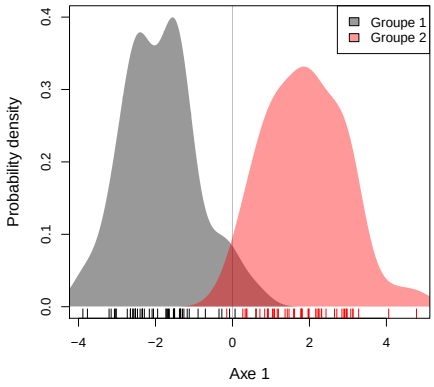
Cette prédiction se fait en trois étapes :

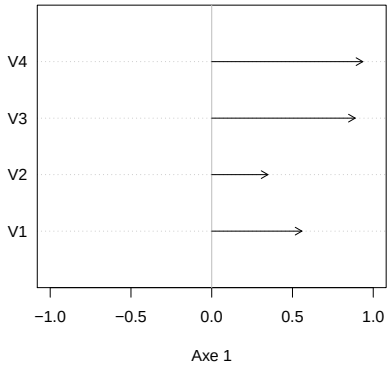
1. Générer une série de sous-modèles à partir du jeu de données initial, par validation croisée.
2. Créer un tableau contenant une colonne par variable quantitative (les noms de colonnes doivent être *strictement identiques* aux noms des variables du jeu de données initial), et remplir chaque ligne en fonction des valeurs pour lesquelles la prédiction doit être faite (il y a donc une prédiction par ligne de ce tableau). Si le jeu de données initial avait été standardisé, il faut également standardiser le jeu de données servant à la prédiction. Pour cela : `new.tab<-stand(new.tab,tableau)`[1] où `new.tab` est le tableau des individus à classer.
3. Réaliser la prédiction : `predict(LDA.vc,new.tab)` où `LDA.vc` est le résultat de la validation croisée (étape 1). La fonction renvoie pour chaque ligne de `new.tab` le groupe prédit (colonne `Group`) et la probabilité de cette prédiction (colonne `Proba`).

Remarque : tout l'intérêt de générer un grand nombre de sous-modèles au moment de la validation croisée est de pouvoir associer à chaque prédiction une probabilité, car chaque sous-modèle va en fait servir à faire sa propre prédiction, ce qui permet d'estimer la fiabilité de la prédiction « moyenne ».


Groupe 3
Groupe 2
Groupe 1
Axe 2
Axe 1


V2
V4
V1
V3
Axe 2
Axe 1


Groupe 1
Groupe 2
Probability density
0.0
0.1
0.2
0.3
0.4
−4
−2
0
2
4
Axe 1


V4
V3
V2
V1
−1.0
−0.5
0.0
0.5
1.0
Axe 1

106. La régression PLS discriminante

— PACKAGE(S) ADDITIONNEL(S) UTILISÉ(S) DANS CETTE FICHE —

[1]`RVAideMemoire`, [2]`pls`

SYNONYMES, TRADUCTIONS ET ABRÉVIATIONS

Français – Régression PLS discriminante
Régression des moindres carrés partiels discriminante

Anglais – *Partial least squares discriminant analysis* (*PLS-DA*)
Projection to latent structures discriminant analysis (*PLS-DA*)

Préparation des données

Il est fortement recommandé de standardiser les variables quantitatives (voir fiche **88**). Une transformation préalable est également souvent bénéfique, pour rendre plus linéaires les relations entre ces variables. Dans tous les cas, le tableau de variables quantitatives doit être transformé en matrice, par exemple *via* `tableau<-as.matrix(tableau)` où `tableau` est le tableau de variables quantitatives.

Remarque : si les variables ont été standardisées (voir fiche **88**), le résultat de la standardisation est déjà une matrice.

Le facteur doit également être transformé en *variables indicatrices*. Pour cela : `var.ind<-dummy(facteur)`[1] où `facteur` est le facteur définissant les groupes.

Réalisation de l'analyse

Pour réaliser la PLS-DA : `PLSDA<-cppls(var.ind~tableau)`[2].

Remarque : par défaut autant d'axes sont créés qu'il y a de variables quantitatives, ce qui peut parfois générer une erreur. Si tel est le cas, réduire le nombre d'axes en ajoutant l'argument `ncomp=10` (10 axes sont largement plus que nécessaire pour la suite).

Qualité de l'analyse

Pourcentage d'erreur de classification

Une des fonctions de toute analyse discriminante est de faire de la *prédiction*, *i.e.* de prédire le groupe auquel appartient un individu dont on connaît seulement la valeur pour les variables quantitatives. Une manière d'estimer la qualité d'une analyse discriminante est donc de tester à quel point elle permet de classer un individu de groupe inconnu sans erreur. On utilise pour cela une méthode de *validation croisée*, qui va générer plusieurs sous-modèles chacun sur une partie du jeu de données (définie aléatoirement) et prédire le groupe des individus non pris en compte dans le modèle. Pour plus de fiabilité l'ensemble de la procédure peut être répété plusieurs fois (avec à chaque fois un découpage aléatoire du jeu de données). Dans le cas d'une PLS-DA les choses sont toutefois assez complexes, et une procédure de *double validation croisée* (*cross model validation* ou 2CV) doit être utilisée, qui consiste en une validation croisée dont à chaque étape on réalise une seconde validation croisée interne (la validation croisée « interne » étant appelée *inner loop*, tandis que la validation croisée « externe » est appelée *outer loop*).

Pour réaliser l'opération : `MVA.cmv(tableau,facteur,model="PPLS-DA",crit.inn="NMC")`[1]. Par défaut le découpage du jeu de données se fait en 7 parties au niveau de la boucle externe (*7-fold cross-validation*) et en 6 parties au niveau de la boucle interne (*6-fold cross-validation*). Ces chiffres peuvent être modifiés grâce aux arguments `kout` et `kinn`. La fonction peut également ajuster ces chiffres automatiquement si au moins un groupe contient moins de 7 individus. Par défaut l'ensemble de la procédure est répété 10 fois (argument `repet`), ce qui au final génère $7 \times 10 = 70$ sous-modèles.

Remarque 1 : si g est le nombre de groupes, on utilise souvent $g-1$ axes pour tester et interpréter une PLS-DA, car c'est théoriquement suffisant. Pour suivre ce principe, ajouter l'argument `ncomp=nb` où `nb` $= g-1$.

Remarque 2 : il existe en fait plusieurs versions de la PLS-DA. La PPLS-DA (*Powered PLS-DA*) en est la plus efficace, c'est celle qui est recommandée et utilisée dans cette fiche.

Si l'on souhaite utiliser la PLS-DA dans un but prédictif, stocker le résultat de la fonction `MVA.cmv()`[1] dans un objet.

Test(s)

Le test à réaliser est un test par permutation basé sur la double validation croisée. Pour le réaliser : `MVA.test(tableau,facteur,model="PPLS-DA",cmv=TRUE)`[1]. Les arguments `kout`, `kinn` et surtout `ncomp` sont disponibles. Le temps de calcul nécessaire pour ce test est relativement long.

Si le facteur a un effet significatif et qu'il y a plus de deux groupes, il est nécessaire de réaliser des comparaisons multiples pour identifier les modalités qui diffèrent. Pour cela : `pairwise.MVA.test(tableau,facteur,model="PPLS-DA",cmv=TRUE)`[1] (avec toujours les arguments `kout`, `kinn` et surtout `ncomp`). Le temps de calcul nécessaire pour réaliser toutes les comparaisons deux-à-deux est très long.

Remarque : avec une PLS-DA, il est *absolument indispensable* de tester si le facteur a un effet significatif avant toute interprétation. En effet, les représentations graphiques auront toujours tendance à montrer des groupes séparés *même si cette séparation est totalement aléatoire* (le phénomène est d'autant plus probable que les variables quantitatives sont bien plus nombreuses que les individus). On ne peut donc *pas* se fier à ces représentations graphiques si les groupes ne sont pas significativement différents.

Représentations graphiques

En PLS-DA on a deux représentations possibles : le *graphe des individus* qui montre la position de ces individus sur un *plan factoriel* composé de deux axes, et le *cercle des corrélations* (voir fiche **89**) qui permet d'interpréter la répartition des individus.

Graphe des individus

Pour tracer le graphe : `MVA.plot(PLSDA,fac=facteur)`[2]. Les axes 1 (horizontal) et 2 (vertical) sont représentés par défaut, ils peuvent être changés grâce aux arguments `xax` et `yax`.

Les arguments `col`, `pch`, `fac.lab`, `contours`, `stars` et `barycenters` permettent de personnaliser la représentation. Voir `?MVA.scoreplot` pour bien d'autres d'options graphiques.

Cercle des corrélations

Pour tracer le graphe : `MVA.plot(PLSDA,"corr")`[2].

Pour supprimer les flèches, ajouter l'argument `arrows=FALSE`. Voir `?MVA.corplot` pour bien d'autres d'options graphiques.

Interprétation

On part du principe que les groupes sont significativement différents. Dans le cas contraire l'interprétation *doit être évitée*.

Le graphe des individus permet d'identifier comment les groupes se structurent, *i.e.* si tout ou partie des groupes se séparent ou au contraire se chevauchent.

Dans un second temps, le cercle des corrélations permet d'identifier les variables qui différencient ces groupes. Pour cela, on repère quelles sont les directions pertinentes pour l'interprétation biologique sur le graphe des individus (ce peuvent être des axes ou n'importe quelles diagonales), et on identifie les variables qui corrèlent le plus avec ces directions sur le cercle des corrélations (voir fiche **89**).

Prédiction

L'un des intérêts des analyses discriminantes est à la fois de comprendre comment les groupes se différencient, mais également de pouvoir prédire le groupe d'un individu pour des valeurs *connues* des variables quantitatives. Prédire un groupe nécessite donc de fixer la valeur de *toutes* les variables quantitatives.

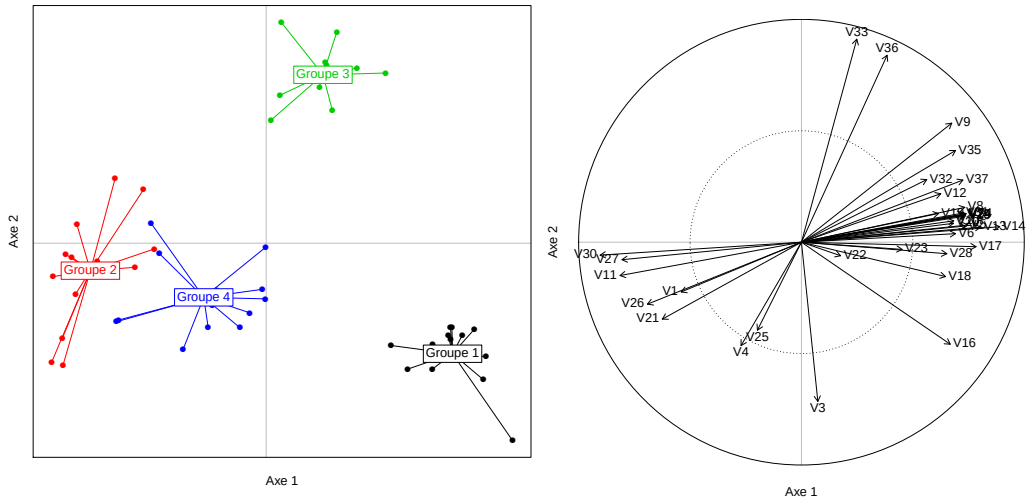
Cette prédiction se fait en trois étapes :

1. Générer une série de sous-modèles à partir du jeu de données initial, par double validation croisée.
2. Créer un tableau contenant une colonne par variable quantitative (les noms de colonnes doivent être *strictement identiques* aux noms des variables du jeu de données initial), et remplir chaque ligne en fonction des valeurs pour lesquelles la prédiction doit être faite (il y a donc une prédiction par ligne de ce tableau). Si le jeu de données initial avait été standardisé, il faut également standardiser

le jeu de données servant à la prédiction. Pour cela : `new.tab<-stand(new.tab,tableau)`[1] où `new.tab` est le tableau des individus à classer.

3. Réaliser la prédiction : `predict(PLSDA.vc,new.tab)` où `PLSDA.vc` est le résultat de la doube validation croisée (étape 1). La fonction renvoie pour chaque ligne de `new.tab` le groupe prédit (colonne `Group`) et la probabilité de cette prédiction (colonne `Proba`).

Remarque : tout l'intérêt de générer un grand nombre de sous-modèles au moment de la double validation croisée est de pouvoir associer à chaque prédiction une probabilité, car chaque sous-modèle va en fait servir à faire sa propre prédiction, ce qui permet d'estimer la fiabilité de la prédiction « moyenne ».



107. L'analyse canonique des correspondances

— PACKAGE(S) ADDITIONNEL(S) UTILISÉ(S) DANS CETTE FICHE —

[1]`vegan`, [2]`RVAideMemoire`

SYNONYMES, TRADUCTIONS ET ABRÉVIATIONS

Français – Analyse canonique des correspondances
Analyse factorielle des correspondances sur variables instrumentales (AFCVI)

Anglais – *Canonical correspondence analysis* (*CCA*)
Constrained correspondence analysis (*CCA*)
Correspondence analysis with respect to instrumental variables (*CAIV*)

Préparation des données

Dans un tableau de contingence ou un tableau de présence-absence, les lignes et les colonnes ont généralement un rôle symétrique (*i.e.* il n'y a pas d'« individus » et de « variables »). Cependant la CCA impose que les entités pour lesquelles les variables explicatives sont définies soient placées en *lignes*, comme dans un tableau classique. On appellera donc les lignes « individus ».

Réalisation de l'analyse

Pour réaliser la CCA : `CCA<-cca(formule,data=tab.explicatif)`[1] où `tab.explicatif` est le tableau contenant les variables explicatives. Voir fiche **40** pour une explication détaillée de la construction d'une formule. Dans cette formule, la réponse est le tableau à expliquer.

Capacité explicative globale

La CCA consiste en fait en deux étapes :

1. Séparer la variation (du tableau à expliquer) due aux variables explicatives (appelée variation *contrainte*) de la variation non expliquée (dite *résiduelle* ou *non contrainte*). La CCA travaille sur une certaine information qui est la *correspondance* entre les lignes et les colonnes du jeu de données, que l'on appellera plus généralement *inertie* (l'inertie est en fait un terme générique, le fait qu'elle représente une correspondance est un cas particulier). À ce stade le tableau à expliquer est considéré asymétriquement puisque les lignes jouent le rôle d'individus.
2. Réaliser deux AFC séparées, l'une sur la variation contrainte (« AFC contrainte ») et l'autre sur la variation non contrainte (« AFC non contrainte »). Comme en AFC classique (voir fiche **98**), lignes et colonnes jouent cette fois un rôle symétrique.

On peut estimer la capacité explicative globale de la CCA grâce au pourcentage d'inertie contrainte de l'analyse (*i.e.* d'inertie du tableau à expliquer expliquée par les variables explicatives). Plus ce pourcentage est élevé et plus la variation observée dans le tableau à expliquer est liée aux variables explicatives. Ce pourcentage est obtenu *via* `MVA.synt(CCA)`[2], dans le premier tableau renvoyé par la fonction.

Qualité de l'analyse

Test(s)

L'effet des variables explicatives est testé *via* `MVA.anova(CCA)`[2]. Un test F par permutation est réalisé.

Si au moins une variable explicative a un effet significatif, on peut se baser sur les résultats de l'AFC *contrainte* pour l'interprétation. Si aucune variable explicative n'a d'effet significatif, interpréter les résultats de cette AFC n'a pas beaucoup d'intérêt puisqu'aucun effet n'est montré.

En lien avec cette AFC contrainte, on peut réaliser des comparaisons multiples entre modalités d'un facteur (ou combinaisons de modalités d'une interaction entre facteurs) à effet significatif. Pour réaliser le test : `pairwise.factorfit(CCA,facteur)`[2] où `facteur` est le facteur d'intérêt.

Synthèse

Si au moins une variable explicative a un effet significatif, on s'intéresse à l'AFC contrainte. Comme pour une AFC classique (voir fiche **98**), on estime la qualité de cette analyse par le pourcentage d'inertie expliqué par chaque axe. Ces pourcentages sont obtenus *via* `MVA.synt(CCA)`[2], dans le deuxième tableau renvoyé par la fonction.

Remarque 1 : il s'agit ici de pourcentages d'inertie *contrainte*, pas totale comme en AFC classique.

Remarque 2 : les pourcentages d'inertie sont toujours en ordre décroissant (*i.e.* l'axe 1 explique plus d'inertie que l'axe 2, qui en explique lui-même plus que l'axe 3...).

Remarque 3 : il n'y a pas de règle absolue sur le nombre d'axes à retenir pour l'interprétation. Il s'agit toujours d'un compromis entre une bonne synthèse de l'information (qui augmente avec le nombre d'axes) et une facilité à interpréter (qui diminue avec le nombre d'axes).

Représentations graphiques

À partir du moment où il y a au moins une variable explicative quantitative, deux représentations sont possibles : le « graphe d'association » et le *cercle des corrélations* (voir fiche **89**). S'il n'y a aucune variable explicative quantitative le seul graphe possible est celui d'association.

Graphe d'association

Sur ce graphe les lignes et les colonnes sont représentées chacune par un point. Il est cependant impossible de représenter à la fois les distances interlignes et les distances intercolonnes sans biais sur le même graphe. Il faut donc choisir entre représenter sans biais les premières (échelle de type 1) ou les secondes (échelle de type 2).

Pour représenter les distances interlignes sans biais : `MVA.plot(CCA,points=FALSE,scaling=1)`[2]. Pour représenter les distances intercolonnes sans biais : `MVA.plot(CCA,points=FALSE,scaling=2)`[2]. Les axes 1 (horizontal) et 2 (vertical) sont représentés par défaut, ils peuvent être changés grâce aux arguments `xax` et `yax`. Par défaut c'est l'AFC contrainte qui est représentée. Pour représenter l'AFC non contrainte, ajouter l'argument `space=2`. Les arguments `col`, `pch` et `points` permettent de personnaliser une telle représentation. Voir `?MVA.scoreplot` pour bien d'autres d'options graphiques.

Ajouter des groupes sur le graphe est possible mais la procédure est plus complexe :

```
> assoc <- MVA.plot(CCA,points=FALSE,col=couleurs)
```
[2] où `couleurs` est un vecteur à deux valeurs, la première pour les *colonnes* et la seconde pour les *lignes*. S'il l'on souhaite afficher des groupes de colonnes, la première couleur doit être `"white"` ; pour des groupes de lignes la seconde couleur doit être `"white"`.

```
> par(new=TRUE)
> MVA.plot(CCA,points=FALSE,xlim=assoc$xlim,ylim=assoc$ylim,set=nb,fac=facteur)
```
[2] où `nb` vaut 1 pour des groupes de lignes ou 2 pour des groupes de colonnes, et `facteur` est le facteur définissant le groupe de chaque ligne/colonne. Les arguments `col`, `pch`, `fac.lab`, `contours`, `stars` et `barycenters` permettent de personnaliser une telle représentation. Voir `?MVA.scoreplot` pour bien d'autres d'options graphiques.

Cercle des corrélations

Pour tracer le graphe : `MVA.plot(CCA,"corr")`[2]. Comme pour le graphe des individus c'est l'AFC contrainte qui est représentée. L'AFC non contrainte n'a pas de sens puisqu'elle ne concerne pas les variables explicatives.

Pour supprimer les flèches, ajouter l'argument `arrows=FALSE`. Voir `?MVA.corplot` pour bien d'autres d'options graphiques.

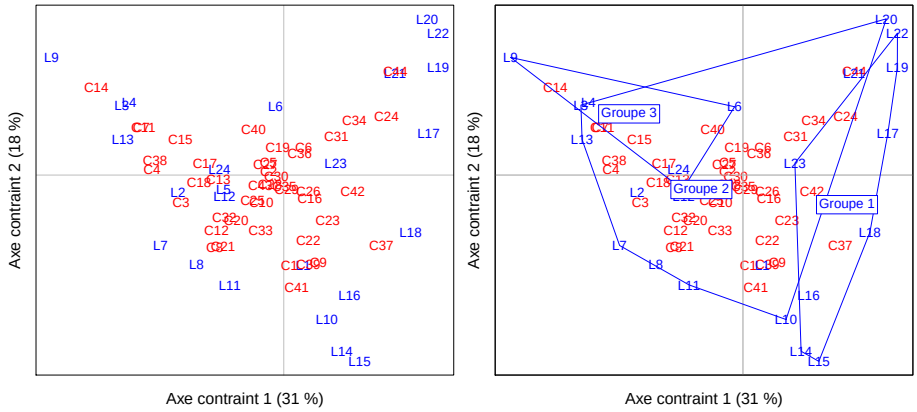
Interprétation

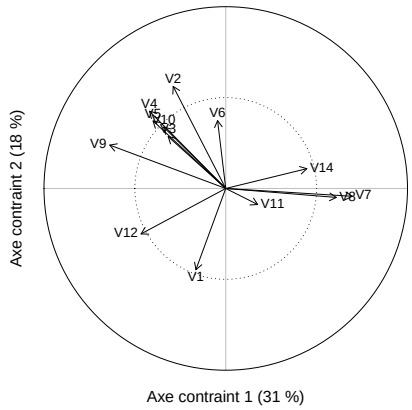
On ne considère que l'AFC contrainte puisque par définition c'est la seule qui permette d'interpréter les résultats en lien avec les variables explicatives.

Le graphe des individus permet (i) d'identifier les associations entre lignes et colonnes du tableau à expliquer (de la même façon qu'en AFC, voir fiche **98**), (ii) d'identifier comment les modalités d'un facteur à effet significatif se répartiessent et (iii) d'identifier des gradients linéaires.

Dans un second temps, le cercle des corrélations permet d'identifier les variables explicatives quantitatives (i) qui différencient d'éventuels groupes et/ou (ii) qui expliquent d'éventuels gradients. Pour

cela, on repère quelles sont les directions pertinentes pour l'interprétation biologique sur le graphe d'association (ce peuvent être des axes ou n'importe quelles diagonales), et on identifie les variables qui corrèlent le plus avec ces directions sur le cercle des corrélations (voir fiche **89**).





108. L'analyse des correspondances discriminante

— PACKAGE(S) ADDITIONNEL(S) UTILISÉ(S) DANS CETTE FICHE —

[1]`ade4`, [2]`RVAideMemoire`

SYNONYMES, TRADUCTIONS ET ABRÉVIATIONS

Français – Analyse des correspondances discriminante
Analyse discriminante des correspondances

Anglais – *Correspondence discriminant analysis* (*CDA*)
Discriminant correspondence analysis (*DCA*)

Préparation des données

Dans un tableau de contingence ou un tableau de présence-absence, les lignes et les colonnes ont généralement un rôle symétrique (*i.e.* il n'y a pas d'« individus » et de « variables »). Cependant la CDA impose que les entités pour lesquelles les groupes sont définis soient placées en *lignes*, comme dans un tableau classique. On appellera donc les lignes « individus » et les colonnes « variables ».

Remarque : la CDA consiste en fait en deux étapes. La première est de réaliser une AFC sur le tableau de contingence (ou de présence-absence) ; le tableau est bien considéré symétriquement à ce moment là (voir fiche **98**). Dans un second temps, les coordonnées des *lignes* du tableau de contingence sur les axes de l'AFC sont utilisées pour réaliser une LDA (voir fiche **105**), dans laquelle les *colonnes* du tableau de contingence vont servir de variables pour l'interprétation.

Réalisation de l'analyse

Pour réaliser la CDA : `CDA<-discrimin.coa(tableau,facteur,scannf=FALSE,nf=nlevels(facteur)-1)`[1] où `tableau` est le tableau de contingence (ou de présence-absence) et `facteur` le facteur définissant les groupes.

Qualité de l'analyse

Pourcentage d'erreur de classification

Une des fonctions de toute analyse discriminante est de faire de la *prédiction*, *i.e.* de prédire le groupe auquel appartient un individu dont on connaît seulement la valeur pour les variables quantitatives (*i.e.* les colonnes du tableau de contingence). Une manière d'estimer la qualité d'une analyse discriminante est donc de tester à quel point elle permet de classer un individu de groupe inconnu sans erreur. On utilise pour cela une méthode de *validation croisée*, qui va générer plusieurs sous-modèles chacun sur une partie du jeu de données (définie aléatoirement) et prédire le groupe des individus non pris en compte dans le modèle. Pour plus de fiabilité l'ensemble de la procédure peut être répété plusieurs fois (avec à chaque fois un découpage aléatoire du jeu de données).

Pour réaliser la validation croisée : `CDA.cv(tableau,facteur)`[2]. Par défaut la fonction découpe le jeu de données en 7 parties (*7-fold cross-validation*), mais ce chiffre peut être modifié *via* l'argument `k=nb` où `nb` est le nombre de sous-jeux de données à générer. La fonction peut également ajuster ce chiffre automatiquement si au moins un groupe contient moins de 7 individus. Par défaut l'ensemble de la procédure est répété 10 fois (argument `repet`), ce qui au final génère $7 \times 10 = 70$ sous-modèles.

Si l'on souhaite utiliser la CDA dans un but prédictif, stocker le résultat de la fonction `CDA.cv()`[2] dans un objet.

Test(s)

Le test à réaliser repose sur les coordonnées des individus sur les axes de l'AFC intermédiaire. Pour le réaliser : `CDA.test(tableau,facteur)`. Si g est le nombre de groupes, par défaut $g-1$ axes de l'AFC sont retenus pour le test. Ce nombre d'axes peut être modifié grâce à l'argument `ncomp=nb` où `nb` est le nombre d'axes souhaité.

Si un seul axe est retenu (par choix ou parce qu'il n'y a que deux groupes), le test est une ANOVA. Si au moins deux axes sont retenus, le test est une MANOVA (*Multivariate ANalysis Of VAriance*), une extenstion de l'ANOVA au cas multivarié.

Si le facteur a un effet significatif et qu'il y a plus de deux groupes, il est nécessaire de réaliser des comparaisons multiples pour identifier les modalités qui diffèrent. Pour cela : `pairwise.CDA.test(tableau, facteur)` (avec toujours l'argument `ncomp`).

Remarque : si le facteur n'a pas d'effet significatif, l'interprétation des résultats de la CDA n'a pas vraiment de sens et toute tentative de prédiction est inutile.

Représentations graphiques

En CDA on a deux représentations possibles : le *graphe des individus* qui montre la position de ces individus sur un *plan factoriel* composé de deux axes, et le *cercle des corrélations* (voir fiche **89**) qui permet d'interpréter la répartition des individus.

Remarque : s'il n'y a que deux groupes, la CDA ne produit qu'un seul axe. Le graphe des individus et le cercle des corrélations se réduisent donc à une seule dimension.

Graphe des individus

Pour tracer le graphe : `MVA.plot(CDA,fac=facteur)`[2]. Les axes 1 (horizontal) et 2 (vertical) sont représentés par défaut (s'il y a au moins deux axes), ils peuvent être changés grâce aux arguments `xax` et `yax`.

Dans le cas d'un graphe à deux dimensions, les arguments `col`, `pch`, `fac.lab`, `contours`, `stars` et `barycenters` permettent de personnaliser la représentation. Dans le cas d'un graphe à une seule dimension, les arguments `col`, `legend` et `legend.lab` sont intéressants. Voir `?MVA.scoreplot` pour bien d'autres d'options graphiques.

Cercle des corrélations

Pour tracer le graphe : `MVA.plot(CDA,"corr")`[2].

Pour supprimer les flèches, ajouter l'argument `arrows=FALSE`. Voir `?MVA.corplot` pour bien d'autres d'options graphiques.

Interprétation

Le graphe des individus permet d'identifier comment les groupes se structurent, *i.e.* si tout ou partie des groupes se séparent ou au contraire se chevauchent.

Dans un second temps, le cercle des corrélations permet d'identifier les variables (*i.e.* les colonnes du tableau de contingence) qui différencient ces groupes. Pour cela, on repère quelles sont les directions pertinentes pour l'interprétation biologique sur le graphe des individus (ce peuvent être des axes ou n'importe quelles diagonales), et on identifie les variables qui corrèlent le plus avec ces directions sur le cercle des corrélations (voir fiche **89**).

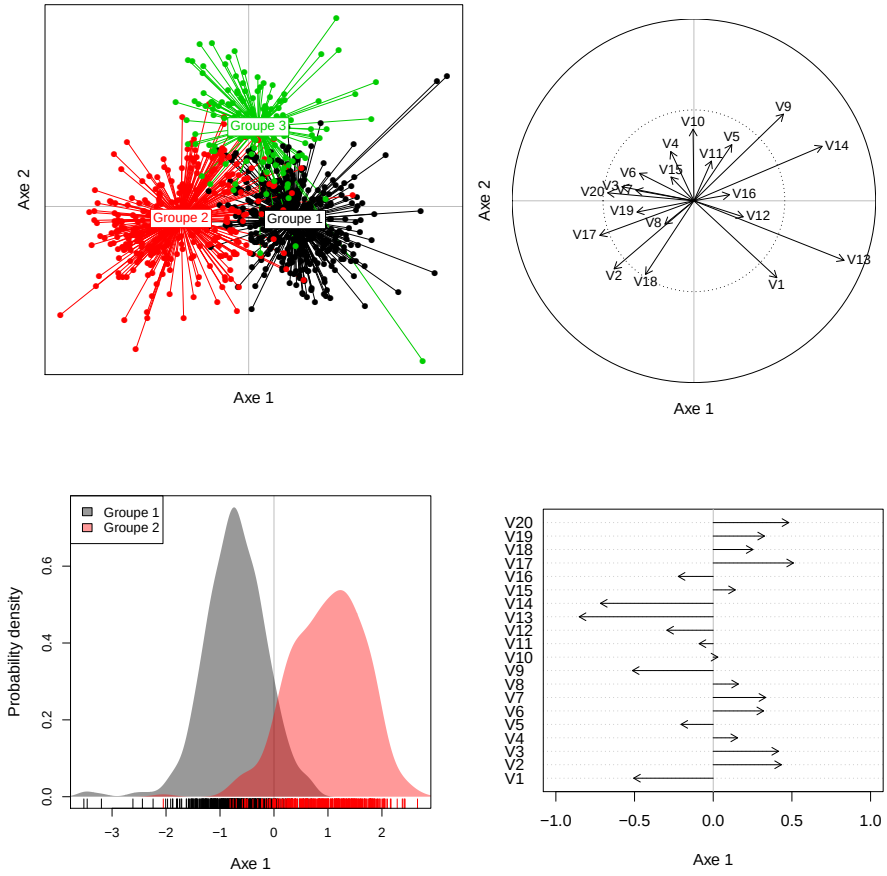
Prédiction

L'un des intérêts des analyses discriminantes est à la fois de comprendre comment les groupes se différencient, mais également de pouvoir prédire le groupe d'un individu pour des valeurs *connues* des variables quantitatives. Prédire un groupe nécessite donc de fixer la valeur de *toutes* les variables quantitatives.

Cette prédiction se fait en trois étapes :

1. Générer une série de sous-modèles à partir du jeu de données initial, par validation croisée.
2. Créer un tableau contenant une colonne par colonne du tableau initial (les noms de colonnes doivent être *strictement identiques* à ceux du jeu de données initial), et remplir chaque ligne en fonction des valeurs pour lesquelles la prédiction doit être faite (il y a donc une prédiction par ligne de ce tableau).
3. Réaliser la prédiction : `predict(CDA.vc,new.tab)` où `CDA.vc` est le résultat de la validation croisée (étape 1) et `new.tab` le tableau des individus à classer. La fonction renvoie pour chaque ligne de `new.tab` le groupe prédit (colonne `Group`) et la probabilité de cette prédiction (colonne `Proba`).

Remarque : tout l'intérêt de générer un grand nombre de sous-modèles au moment de la validation croisée est de pouvoir associer à chaque prédiction une probabilité, car chaque sous-modèle va en fait servir à faire sa propre prédiction, ce qui permet d'estimer la fiabilité de la prédiction « moyenne ».



109. Analyser un ensemble de variables quantitatives

— PACKAGE(S) ADDITIONNEL(S) UTILISÉ(S) DANS CETTE FICHE —

[1]`vegan`, [2]`RVAideMemoire`, [3]`car`

⚠ L'analyse présentée dans cette fiche est basée sur un modèle. Il est indispensable de maîtriser les notions des fiches **39** à **42**. ⚠

Réponse

Le modèle qui va être utilisé fait l'hypothèse que les matrices de variance-covariance (l'équivalent multivarié de la variance) sont homogènes entre les différentes modalités des variables explicatives qualitatives. Pour le tester : `anova(betadisper(dist(tableau),facteur)`[1]`)` où `tableau` est le tableau *à expliquer* et `facteur` le facteur définissant les groupes. Le modèle est toutefois assez robuste à un non respect (modéré) de cette condition. Si elle n'est pas du tout respectée, une transformation préalable du tableau peut grandement aider à améliorer la situation (voir fiche **88**).

Dans tous les cas, les variables à expliquer doivent être bien moins nombreuses que les individus et elles ne doivent pas être trop corrélées entre elles. Si cette double condition n'est pas respectée, transformer le tableau à expliquer en matrice de distance (voir fiche **101**) et utiliser cette matrice comme réponse dans une analyse dédiée (voir fiche **111**).

Après avoir fait ces vérifications et possiblement transformé le tableau à expliquer, celui-ci doit être converti en matrice. Pour cela : `tableau<-as.matrix(tableau)`.

Modèle utilisé

Le modèle utilisé est un Modèle Linéaire Multivarié (*Multivariate Linear Model* ou MLM), une extension du Modèle Linéaire classique du cadre univarié (voir fiche **76**). À la différence du Modèle Linéaire classique, il y a plusieurs variables à expliquer dans le MLM.

Construction du modèle

Pour créer le modèle : `modele<-lm(formule)`. Voir fiche **40** pour une explication détaillée de la construction d'une formule. Dans cette formule, la réponse est `tableau` (*i.e.* le tableau à expliquer). De manière générale, on peut dire que :

- inclure un facteur permet de tester si la position des individus dans l'espace multidimensionnel des variables à expliquer diffère entre les niveaux de ce facteur.
- inclure une covariable permet de tester s'il existe une relation entre cette covariable et les variables à expliquer.
- inclure une interaction entre deux variables explicatives permet de tester si l'effet de l'une dépend de l'autre (le raisonnement est le même pour une interaction incluant plus de deux termes). Le cas particulier d'une interaction entre une covariable et un facteur permet de tester si la relation entre la covariable et les variables à expliquer est différente selon la modalité du facteur.

Remarque : bien que la gestion des covariables ne pose pas de problème théorique avec les MLMs, il est en pratique difficile de se représenter leur effet dans le cadre multivarié (sauf à l'interpréter séparément pour chaque variable à expliquer). Ce type de modèle se prête par contre très bien à l'analyse de l'effet de facteurs.

Vérification de l'ajustement aux données

Avant d'aller plus loin, il est indispensable de vérifier que le modèle s'ajuste bien aux données. Cette étape est fondamentale, et ce pour tout modèle, car un test basé sur un modèle mal ajusté n'est simplement pas fiable (ni valide). L'ajustement d'un MLM est cependant plus difficile à vérifier que celui d'un modèle à réponse univariée. Le seul critère réellement observable est la distribution des *résidus* du modèle (*i.e.* les écarts entre les valeurs réellement observées et celles prédites par le modèle). Pour que le

modèle soit validé, les résidus doivent suivre une distribution normale multivariée (une extension de la distribution normale classique).

Pour faire cette vérification : `plotresid(modele)`[2]. L'hypothèse de normalité est acceptée lorsque les points sont à peu près alignés sur une droite. Ces points peuvent ne pas être parfaitement alignés sur la droite, mais s'ils restent à peu près dans l'intervalle de confiance de celle-ci (représenté en pointillés), l'ajustement est considéré comme correct.

Si l'hypothèse de normalité est intenable, l'alternative au MLM est de transformer l'ensemble des variables à expliquer en une matrice de distance (voir fiche **101**) et d'utiliser cette matrice comme réponse dans une analyse dédiée (voir fiche **111**).

Test

L'effet des variables explicatives est testé par une MANOVA (*Multivariate ANalysis Of VAriance*), une extension de l'ANOVA au cas multivarié. Pour réaliser le test : `Manova(modele)`[3]. Le test réalisé est de type II (voir fiche **42** pour une explication détaillée des hypothèses testées).

Si un facteur (ou une interaction entre plusieurs facteurs) a un effet significatif, il est nécessaire de réaliser des comparaisons multiples pour identifier les modalités (ou combinaisons de modalités) qui différent. Voir fiche **43** pour réaliser ces comparaisons.

Si une covariable a une effet significatif, il n'y a pas vraiment d'autre solution pour estimer la direction de son effet que de l'étudier séparément sur chaque variable à expliquer. Cette direction est donnée par le *signe* du paramètre associé. Les valeurs de tous les paramètres des différents modèles (un par variable à expliquer) sont obtenues *via* `summary(modele)`. Elles sont appelées `Estimate` et se trouvent dans les tableaux `Coefficients`. Si le coefficient portant le nom de la covariable est négatif, la valeur de la variable à expliquer correspondante diminue quand la valeur de la covariable augmente ; s'il est positif, la valeur de la variable à expliquer augmente quand la valeur de la covariable augmente.

Prédiction à partir du modèle

L'intérêt des modèles est à la fois de comprendre l'importance des différentes variables explicatives sur les variables à expliquer, mais également de prédire les valeurs que prendraient ces variables à expliquer pour des valeurs *connues* des variables explicatives. Faire une prédiction (une par variable à expliquer) nécessite donc de fixer la valeur de *toutes* les variables explicatives.

Deux méthodes peuvent être utilisées pour la prédiction, les deux étant basées sur la fonction `predict()` :

— donner la valeur (ou les valeurs pour faire plusieurs prédictions) de chacune des variables explicatives directement dans la fonction, sous la forme d'une liste : `predict(modele,newdata=list(variables))`, où `variables` est un enchaînement de `variable1=valeur,variable2=valeur...`

— créer un tableau contenant une colonne par variable explicative (les noms de colonnes doivent être *strictement identiques* aux noms des variables du modèle), et remplir chaque ligne en fonction des valeurs pour lesquelles la prédiction doit être faite (il y a donc une prédiction par ligne de ce tableau). Puis : `predict(modele,newdata=tab.pred)`.

EXEMPLE(S)

Avec un modèle contenant un facteur à deux niveaux (A et B), une covariable variant de 0 à 30, et leur interaction :

```
> modele <- lm(reponse~facteur*covariable)
```

On peut prédire la valeur de chaque variable à expliquer de cette façon :

```
> predict(modele,newdata=list(facteur="A",covariable=10))
```

Ou, pour plusieurs prédictions :

```
> predict(modele,newdata=list(facteur=c("A","B"),covariable=c(10,10)))
```

Ou encore créer un tableau de ce type :

```
> tab.pred
  facteur covariable
1       A         10
2       B         10
```

Puis :

```
> predict(modele,newdata=tab.pred)
```

110. L'analyse de redondance sur matrice de distance

— PACKAGE(S) ADDITIONNEL(S) UTILISÉ(S) DANS CETTE FICHE —

[1]`vegan`, [2]`RVAideMemoire`

SYNONYMES, TRADUCTIONS ET ABRÉVIATIONS

Français – Analyse de redondance sur matrice de distance
Anglais – *Distance-based redundancy analysis* (*db-RDA*)
Canonical analysis of principal coordinates (*CAP*)
Constrained analysis of principal coordinates (*CAP*)

Préparation des données

Il est indispensable de vérifier si la matrice de distance est euclidienne ou non (voir fiche **101**).

Réalisation de l'analyse

Pour réaliser la db-RDA : `dbRDA<-dbrda(formule,data=tab.explicatif)`[1] où `tab.explicatif` est le tableau contenant les variables explicatives. Voir fiche **40** pour une explication détaillée de la construction d'une formule. Dans cette formule, la réponse est `mat.dist` (*i.e.* la matrice de distance). Si celle-ci n'est pas euclidienne, il est nécessaire d'appliquer une correction. Ajouter pour cela l'argument `add=TRUE`.

Capacité explicative globale

La db-RDA consiste en fait en deux étapes :

1. Réaliser une PCoA sur la matrice de distance (voir fiche **102**).
2. Réaliser une RDA (voir fiche **104**) sur les résultats de la PCoA. Le résultat de la RDA consiste en deux ACP (voir fiche **97**) :
   — Une sur la variation (de la matrice de distance) due aux variables explicatives, appelée variation *contrainte*. C'est l'« ACP contrainte ».
   — Une sur la variation non expliquée, appelée variation *résiduelle* ou *non contrainte*. C'est l'« ACP non contrainte ».

On peut estimer la capacité explicative globale de la db-RDA grâce au pourcentage de variance contrainte de l'analyse (*i.e.* de variance de la matrice de distance expliquée par les variables explicatives). Plus ce pourcentage est élevé et plus la variation observée dans la matrice de distance est liée aux variables explicatives. Ce pourcentage est obtenu *via* `MVA.synt(dbRDA)`[2], dans le premier tableau renvoyé par la fonction.

Qualité de l'analyse

Test(s)

L'effet des variables explicatives est testé *via* `MVA.anova(dbRDA)`[2]. Un test F par permutation est réalisé.

Si au moins une variable explicative a un effet significatif, on peut se baser sur les résultats de l'ACP *contrainte* pour l'interprétation. Si aucune variable explicative n'a d'effet significatif, interpréter les résultats de cette ACP n'a pas beaucoup d'intérêt puisqu'aucun effet n'est montré.

En lien avec cette ACP contrainte, on peut réaliser des comparaisons multiples entre modalités d'un facteur (ou combinaisons de modalités d'une interaction entre facteurs) à effet significatif. Pour réaliser le test : `pairwise.factorfit(dbRDA,facteur)`[2] où `facteur` est le facteur d'intérêt.

Synthèse

Si au moins une variable explicative a un effet significatif, on s'intéresse à l'ACP contrainte. Comme pour une ACP classique (voir fiche **97**), on estime la qualité de cette analyse par le pourcentage de variance expliqué par chaque axe. Ces pourcentages sont obtenus *via* `MVA.synt(dbRDA)`[2], dans le deuxième tableau renvoyé par la fonction.

Remarque 1 : il s'agit ici de pourcentages de variance *contrainte*, pas totale comme en ACP classique.

Remarque 2 : les pourcentages de variance sont toujours en ordre décroissant (*i.e.* l'axe 1 explique plus de variance que l'axe 2, qui en explique lui-même plus que l'axe 3...).

Remarque 3 : il n'y a pas de règle absolue sur le nombre d'axes à retenir pour l'interprétation. Il s'agit toujours d'un compromis entre une bonne synthèse de l'information (qui augmente avec le nombre d'axes) et une facilité à interpréter (qui diminue avec le nombre d'axes).

Représentations graphiques

À partir du moment où il y a au moins une variable explicative quantitative, deux représentations sont possibles : le *graphe des individus* qui montre la position de ces individus sur un *plan factoriel* composé de deux axes, et le *cercle des corrélations* (voir fiche **89**). S'il n'y a aucune variable explicative quantitative le seul graphe possible est celui des individus.

Graphe des individus

Pour tracer le graphe : `MVA.plot(dbRDA)`[2]. Les axes 1 (horizontal) et 2 (vertical) sont représentés par défaut, ils peuvent être changés grâce aux arguments `xax` et `yax`. Par défaut c'est l'ACP contrainte qui est représentée. Pour représenter l'ACP non contrainte, ajouter l'argument `space=2`.

Pour ajouter des groupes sur le graphe, utiliser l'argument `fac=facteur` où `facteur` est le facteur définissant le groupe de chaque individu. Les arguments `col`, `pch`, `fac.lab`, `contours`, `stars` et `barycenters` permettent de personnaliser une telle représentation.

Voir `?MVA.scoreplot` pour bien d'autres d'options graphiques.

Cercle des corrélations

Pour tracer le graphe : `MVA.plot(RDA,"corr")`[2]. Comme pour le graphe des individus c'est l'ACP contrainte qui est représentée. L'ACP non contrainte n'a pas de sens puisqu'elle ne concerne pas les variables explicatives.

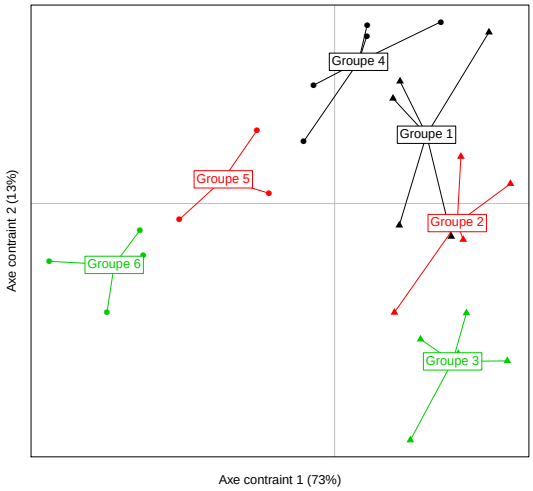
Pour supprimer les flèches, ajouter l'argument `arrows=FALSE`. Voir `?MVA.corplot` pour bien d'autres d'options graphiques.

Interprétation

On ne considère que l'ACP contrainte puisque par définition c'est la seule qui permette d'interpréter les résultats en lien avec les variables explicatives.

Le graphe des individus permet d'identifier la structuration des données de la matrice de distance qui est due aux variables explicatives. On y repère comment les modalités d'un facteur à effet significatif se répartissent, ou des gradients linéaires.

Dans un second temps, le cercle des corrélations permet d'identifier les variables explicatives quantitatives qui expliquent d'éventuels gradients. Pour cela, on repère quelles sont les directions pertinentes pour l'interprétation biologique sur le graphe des individus (ce peuvent être des axes ou n'importe quelles diagonales), et on identifie les variables qui corrèlent le plus avec ces directions sur le cercle des corrélations (voir fiche **89**).


Groupe 4
Groupe 1
Groupe 5
Groupe 2
Groupe 6
Groupe 3
Axe contraint 2 (13%)
Axe contraint 1 (73%)

111. Analyser une matrice de distance

— PACKAGE(S) ADDITIONNEL(S) UTILISÉ(S) DANS CETTE FICHE —

[1]`RVAideMemoire`

Le test à réaliser est non paramétrique et souvent appelé « MANOVA par permutation ». Il est basé sur un modèle (assez analogue au MLM, voir fiche **109**), ce qui permet d'étudier l'effet de plusieurs variables explicatives à la fois (quantitatives et/ou qualitatives), possiblement en interactions. Les *p-value*s sont cependant obtenues par permutation, donc non paramétriquement.

Pour réaliser le test : `adonis.II(formule)`[1]. Voir fiche **40** pour une explication détaillée de la construction d'une formule. Dans cette formule, la réponse est la matrice de distance. Cette matrice n'a pas à être euclidienne.

Si un facteur (ou une interaction entre plusieurs facteurs) a un effet significatif, il est nécessaire de réaliser des comparaisons multiples pour identifier les modalités (ou combinaisons de modalités) qui diffèrent. Pour réaliser ces comparaisons : `pairwise.perm.manova(mat.dist,facteur)`[1] où `mat.dist` est la matrice de distance et `facteur` la facteur d'intérêt.

Si une covariable a un effet significatif, l'interprétation n'est pas si facile. Le plus simple est soit de réaliser une PCoA sur la matrice de distance (voir fiche **102**) puis de corréler ses résultats à la covariable (voir fiche **91**), soit de réaliser une db-RDA (voir fiche **110**). La première approche est préférable si la covariable n'est pas contrôlée, la seconde si elle l'est.

112. L'analyse PLS à deux blocs

— PACKAGE(S) ADDITIONNEL(S) UTILISÉ(S) DANS CETTE FICHE —

[1]`mixOmics`, [2]`RVAideMemoire`

SYNONYMES, TRADUCTIONS ET ABRÉVIATIONS

Français – Analyse PLS à deux blocs
Anglais – *2-block partial least squares* (*2B-PLS*)
2-block projection to latent structures (*2B-PLS*)

Préparation des données

La 2B-PLS fonctionne d'autant mieux que les variables de chaque tableau ont une distribution à peu près normale (au moins symétrique) et qu'elles sont reliées entre elles par des relations à peu près linéaires. Une transformation préalable des données peut grandement aider à améliorer la situation (voir fiche **88**).

Il est également recommandé, la plupart du temps, de standardiser les variables de chaque tableau avant l'analyse (voir fiche **88**). Cela permet de donner le même poids à toutes les variables.

Réalisation de l'analyse

Pour réaliser la 2B-PLS : `PLS2B<-pls(tableau1,tableau2,mode="canonical")`[1] où `tableau1` et `tableau2` sont les deux tableaux. Si l'on souhaite standardiser les variables mais que l'on n'a pas effectué l'opération au préalable, la fonction `pls()`[1] le fait par défaut. Deux paires d'axes sont construits par défaut, pour changer cette valeur ajouter l'argument `ncomp=nb` où `nb` est le nombre de paires d'axes souhaité.

Remarque : on parle de *paires* d'axes en 2B-PLS car l'analyse créé deux ordinations séparées, l'une pour le premier tableau et l'autre pour le second. On a donc un axe 1 pour le premier tableau, et un axe 1 pour le second tableau (et ainsi de suite).

Qualité de l'analyse

Test

La 2B-PLS a pour objectif de synthétiser au mieux une certaine information qui est la *covariance* entre les deux jeux de données. Mais pour que son interprétation soit pertinente, encore faut-il qu'il y ait une covariance (*i.e.* une relation) significative. La significativité de la covariance est testée *via* `cov.test(tableau1,tableau2)`[2]. Un test par permutation est réalisé. Si sa *p-value* est significative, les résultats de la 2B-PLS peuvent être interprétés. Dans le cas contraire l'interprétation n'a pas de sens puisqu'il n'y a pas d'association significative entre les deux tableaux.

Remarque : la covariance est intimement liée à la notion de corrélation (voir fiche **84**). Elle peut donc être négative ou positive, indiquant une relation négative ou positive entre les deux tableaux (ou parties de ces tableaux). Cependant la covariance n'est pas bornée, donc sa valeur absolue n'est pas directement interprétable (d'où l'intérêt de la corrélation, qui est une version standardisée de la covariance).

Synthèse

Si la covariance entre les deux tableaux est significative, on estime la qualité de l'analyse par le pourcentage de covariance expliqué par chaque paire d'axes. Ces pourcentages sont obtenus *via* `MVA.synt(PLS-2B)`[2], dans le premier tableau renvoyé par la fonction.

Remarque 1 : les pourcentages renvoyés par la fonction `MVA.synt()`[2] sont en fait des pourcentages de covariance *au carré*. Peu importe pour l'interprétation : plus le pourcentage est élevé et plus la paire d'axes en question explique une part importante de la covariance entre les jeux de données.

Remarque 2 : les pourcentages de covariance sont toujours en ordre décroissant (*i.e.* la paire d'axes 1 explique plus de covariance que la paire d'axes 2, qui en explique elle-même plus que la paire d'axes 3...).

Remarque 3 : il n'y a pas de règle absolue sur le nombre d'axes à retenir pour l'interprétation. Il s'agit toujours d'un compromis entre une bonne synthèse de l'information des jeux de données (qui augmente avec le nombre d'axes) et une facilité à interpréter (qui diminue avec le nombre d'axes).

Représentations graphiques

En 2B-PLS on a deux représentations possibles *pour chaque tableau* : le *graphe des individus* qui montre la position de ces individus sur un *plan factoriel* composé de deux axes, et le *cercle des corrélations* (voir fiche **89**) qui permet d'interpréter la répartition des individus.

Graphe des individus

Pour tracer le graphe du premier tableau : `MVA.plot(PLS2B,space=1)`[2]. Pour le second tableau : `MVA.plot(PLS2B,space=2)`[2].

Les axes 1 (horizontal) et 2 (vertical) sont représentés par défaut, ils peuvent être changés grâce aux arguments `xax` et `yax`. Voir `?MVA.scoreplot` pour bien d'autres d'options graphiques.

Cercle des corrélations

Pour tracer le graphe du premier tableau : `MVA.plot(PLS2B,"corr",space=1)`[2]. Pour le second tableau : `MVA.plot(PLS2B,"corr",space=2)`[2].

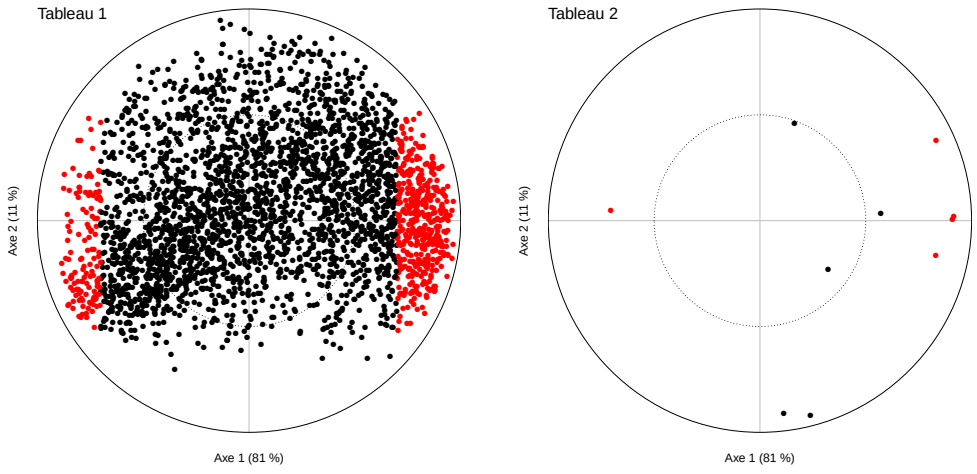
Pour supprimer les flèches, ajouter l'argument `arrows=FALSE`. Voir `?MVA.corplot` pour bien d'autres d'options graphiques.

Interprétation

L'objectif de la 2B-PLS est le plus souvent d'identifier les variables de chaque tableau qui sont associées entre elles, mais aussi (et surtout) les associations entre variables de tableaux différents. À ce titre l'interprétation se base essentiellement sur les cercles des corrélations.

Les pourcentages de covariance expliqués indiquent sur quels axes se concentrer pour l'interprétation. On se place dans le cas le plus fréquent, où une seule paire d'axes est suffisante pour expliquer une grande part de la covariance entre les deux tableaux. Dans un tel cas, on repère pour chaque tableau les variables les plus corrélées au premier axe (voir fiche **89**). Ce sont ces variables qui sont les plus associées.

Au sein d'un tableau, l'interprétation est simple : les variables aux extrémités opposées d'un axe sont corrélées négativement (voir fiche **89**). Pour la relation entre variables de tableaux différents, il manque cependant une information essentielle : dans quel sens sont corrélées les paires d'axes. Ces corrélations sont données par la fonction `MVA.synt()`[2], dans le second tableau qu'elle renvoie. Si la corrélation pour la première paire d'axes est *positive*, les variables dirigées vers la *gauche* de l'axe 1 pour le premier tableau sont positivement reliées aux variables dirigées vers la *gauche* de l'axe 1 pour le second tabeau. Si la corrélation pour la première paire d'axes est *négative*, les variables dirigées vers la *gauche* de l'axe 1 pour le premier tableau sont positivement reliées aux variables dirigées vers la *droite* de l'axe 1 pour le second tabeau.



113. L'analyse procustéenne

— PACKAGE(S) ADDITIONNEL(S) UTILISÉ(S) DANS CETTE FICHE —

[1]`vegan`, [2]`RVAideMemoire`

SYNONYMES, TRADUCTIONS ET ABRÉVIATIONS

Français – Analyse procustéenne
Analyse Procuste
Rotation procustéenne
Rotation Procuste

Anglais – *Procrustes analysis* (*Proc*)
Procrustes rotation (*Proc*)
Procrustean superimposition analysis (*Proc*)

Préparation des données

L'analyse procustéenne est intéressante quand elle est réalisée sur deux tableaux à deux colonnes chacun (voir fiche **115** pour le cas où au moins un tableau contient plus de deux colonnes). Ce peuvent être deux tableaux à deux variables chacun, ou deux axes d'une précédente ordination (voir fiche **90** pour récupérer les coordonnées des individus sur ces axes). Une des applications courantes de l'analyse procustéenne est ainsi de comparer les résultats de deux ordinations réalisées sur une matrice de distance, en particulier deux nMDS (voir fiche **103**).

Réalisation de l'analyse

Pour réaliser l'analyse procustéenne : `Proc<-procrustes(tableau1,tableau2)`[1] où `tableau1` et `tableau2` sont les deux tableaux. Le premier est appelé X et le second Y.

Remarque : l'un et/ou l'autre des deux tableaux peut être directement une ordination si elle a été réalisée avec le package `vegan`.

Qualité de l'analyse

On peut à la fois estimer la « corrélation » entre les deux tableaux et la tester *via* `protest(tableau1,tableau2)`[1]. La fonction renvoie ce qui s'apparente à un coefficient de corrélation (`Correlation in a symmetric Procrustes rotation`) et le résultat d'un test non paramétrique appelé PROTEST (basé sur le coefficient de corrélation). Une *p-value* significative indique qu'il y a une concordance significative entre les deux tableaux.

Représentation graphique

La seule représentation possible en analyse procustéenne est le *graphe des individus*, qui montre la position des individus à la fois pour le tableau X et le tableau Y, sur un *plan factoriel* composé de deux axes. Ces deux axes correspondent aux variables du tableau X.

Pour tracer le graphe : `MVA.plot(Proc,"pairs")`[2]. Chaque individu est représenté par une flèche. Celle-ci démarre aux coordonnées de l'individu pour le tableau X, et s'arrête aux coordonnées de ce même individu pour le tableau Y (celui-ci ayant subi rotation, translation et/ou homothétie pour s'ajuster au mieux au tableau X).

Voir `?MVA.pairplot` pour bien d'autres options graphiques.

Interprétation

Plus les flèches du graphe des individus sont courtes et plus cela indique une concordance forte entre les deux tableaux (*i.e.* les individus sont à peu près positionnés au même endroit quel que soit le tableau). À l'inverse, plus les flèches sont longues et partent dans tous les sens, et plus cela indique une

concordance faible entre les deux tableaux. Il y aura toujours une cohérence entre le graphe et le résultat du test statistique.

La longueur des flèches n'est jamais identique pour tous les individus ; on peut ainsi identifier lesquels sont « semblables » dans les deux tableaux ou au contraire très différents.

114. L'analyse de co-inertie

PACKAGE(S) ADDITIONNEL(S) UTILISÉ(S) DANS CETTE FICHE

[1]RVAideMemoire, [2]ade4

SYNONYMES, TRADUCTIONS ET ABRÉVIATIONS

Français – Analyse de co-inertie
Anglais – *Co-inertia analysis* (*CoIA* ou *CIA*)

Préparation des données

La CIA est une méthode très générale permettant de coupler deux tableaux. Ceux-ci peuvent être de types différents (un tableau de variables classique et un tableau de contingence par exemple), ou contenir des variables de nature différente (quantitatives et/ou qualitatives).

La première étape de l'analyse est de réaliser une ordination séparée sur les deux tableaux. Selon la nature des variables, on utilise le plus souvent l'ACP (voir fiche **97**), l'AFC (voir fiche **98**), l'ACM (voir fiche **99**) ou l'analyse mixte (voir fiche **100**). les deux analyses n'ont pas à être identiques.

Remarque 1 : la CIA exige que les poids des *lignes* des deux tableaux soient égaux entre ces tableaux. Avec l'ACP, l'ACM et l'analyse mixte c'est bien le cas puisque, sauf volonté contraire explicite, toutes les lignes ont le même poids. Avec l'AFC la situation est différente, car dans cette analyse le poids d'une ligne est fonction de la somme des valeurs de cette ligne. Pour coupler une AFC à une autre analyse dans une CIA, il est donc nécessaire de recréer cette autre analyse en spécifiant comme poids ceux utilisés par l'AFC. Cette opération peut se faire automatiquement *via* `analyse.bis<-ord.rw(analyse,AFC)`[1] où `analyse` est l'ordination à coupler avec l'AFC, et `AFC` l'AFC. L'ordination `analyse.bis` est celle qui sera utilisée dans la CIA.

Remarque 2 : pour la raison expliquée dans la remarque précédente, il n'est pas possible de coupler deux AFC dans une CIA.

Réalisation de l'analyse

Pour réaliser la CIA : `CIA<-coinertia(analyse1,analyse2,scannf=FALSE,nf=10)`[2] où `analyse1` et `analyse2` sont les ordinations réalisées séparément sur chaque tableau.

Remarque : la fonction `coinertia()`[2] n'accepte que des ordinations réalisées grâce à des fonctions du package `ade4`. Cela est bien le cas pour l'ACM (voir fiche **99**) et l'analyse mixte (voir fiche **100**), mais pas pour l'ACP (voir fiche **97**) et l'AFC (voir fiche **98**) qui sont réalisées grâce au package `vegan`. Pour convertir le résultat d'une de ces ordinations en objet utilisable par la fonction `coinertia()`[2] : `analyse.bis<-to.dudi(analyse)`[1] où `analyse` est le nom de l'ACP ou l'AFC. Si la fonction `ord.rw()`[1] est utilisée en amont de la CIA, l'ordination qu'elle renvoie est directement compatible avec `coinertia()`[2].

EXEMPLE(S)

On veut coupler dans une CIA une ACP (nommée `ACP`) et une AFC (nommée `AFC`), les deux ayant été réalisées grâce au package `vegan` comme expliqué dans les fiches **97** et **98**. On commence par recréer l'ACP en donnant aux lignes les poids utilisés par l'AFC :

```
> ACP.bis <- ord.rw(ACP,AFC)[1]
```

L'objet `ACP.bis` est directement compatible avec la fonction `coinertia()`[2]. Cependant ce n'est pas le cas de l'objet `AFC`. On transforme celui-ci au bont format :

```
> AFC.bis <- to.dudi(AFC)[1]
```

La CIA peut maintenant être réalisée :

```
> CIA <- coinertia(ACP.bis,AFC.bis,scannf=FALSE,nf=10)[2]
```

Qualité de l'analyse

Test

La CIA a pour objectif de synthétiser au mieux une certaine information qui est la *co-inertie* entre les deux tableaux, *i.e.* la partie de leurs informations respectives qui varie en commun. Mais pour que l'interprétation de la CIA soit pertinente, encore faut-il qu'il y ait une co-inertie (*i.e.* une relation) significative. La significativité de la co-inertie est testée par un test par permutation basé sur le coefficient RV (aucun rapport avec l'auteur de cet aide-mémoire). Ce test est réalisé *via* `randtest(CIA)`[2]. La valeur du coefficient est appelée `Observation` et vaut entre 0 (aucune co-inertie) et 1 (co-inertie parfaite). Une *p-value* significative indique qu'il y a une concordance significative entre les deux tableaux, et donc que les résultats de la CIA peuvent être interprétés. Dans le cas contraire l'interprétation n'a pas de sens puisqu'il n'y a pas de relation significative entre les deux tableaux.

Remarque 1 : si l'une des deux ordinations couplées par la CIA est une AFC, il est nécessaire d'ajouter l'argument `fixed=nb` où `nb` est le numéro de l'AFC (1 si c'est le premier argument de `coinertia()`[2], 2 si c'est le deuxième).

Remarque 2 : à ce jour ce test n'est pas disponible si la CIA inclut une analyse mixte (voie fiche **100**).

Synthèse

Si la co-inertie entre les deux tableaux est significative, on estime la qualité de l'analyse par le pourcentage de co-inertie expliqué par chaque paire d'axes. Ces pourcentages sont obtenus *via* `MVA.synt(CIA)`[1], dans le premier tableau renvoyé par la fonction.

Remarque 1 : on parle de *paires* d'axes en CIA car l'analyse créé deux ordinations séparées (sur la base de celles fournies), l'une pour le premier tableau et l'autre pour le second. On a donc un axe 1 pour le premier tableau, et un axe 1 pour le second tableau (et ainsi de suite).

Remarque 2 : les pourcentages de co-inertie sont toujours en ordre décroissant (*i.e.* la paire d'axes 1 explique plus de co-inertie que la paire d'axes 2, qui en explique elle-même plus que la paire d'axes 3...).

Remarque 3 : il n'y a pas de règle absolue sur le nombre d'axes à retenir pour l'interprétation. Il s'agit toujours d'un compromis entre une bonne synthèse de l'information des jeux de données (qui augmente avec le nombre d'axes) et une facilité à interpréter (qui diminue avec le nombre d'axes).

Représentations graphiques

En CIA on a deux représentations possibles *pour chaque tableau* : le *graphe des individus* qui montre la position de ces individus sur un *plan factoriel* composé de deux axes, et le *cercle des corrélations* (voir fiche **89**) qui permet d'interpréter la répartition des individus.

Remarque : seul le graphe des individus est disponible pour une ACM (voie fiche **99**).

Graphe des individus

Pour tracer le graphe du premier tableau : `MVA.plot(CIA,space=1)`[1]. Pour le second tableau : `MVA.plot(CIA,space=2)`[1].

Les axes 1 (horizontal) et 2 (vertical) sont représentés par défaut, ils peuvent être changés grâce aux arguments `xax` et `yax`.

Pour ajouter des groupes sur le graphe, utiliser l'argument `fac=facteur` où `facteur` est le facteur définissant le groupe de chaque individu. Les arguments `col`, `pch`, `fac.lab`, `contours`, `stars` et `barycenters` permettent de personnaliser une telle représentation.

Voir `?MVA.scoreplot` pour bien d'autres d'options graphiques.

Cercle des corrélations

Pour tracer le graphe du premier tableau : `MVA.plot(CIA,"corr",space=1)`[2]. Pour le second tableau : `MVA.plot(CIA,"corr",space=2)`[2].

Pour supprimer les flèches, ajouter l'argument `arrows=FALSE`. Voir `?MVA.corplot` pour bien d'autres d'options graphiques.

Interprétation

L'objectif de la CIA est d'identifier les associations entre variables de tableaux différents.

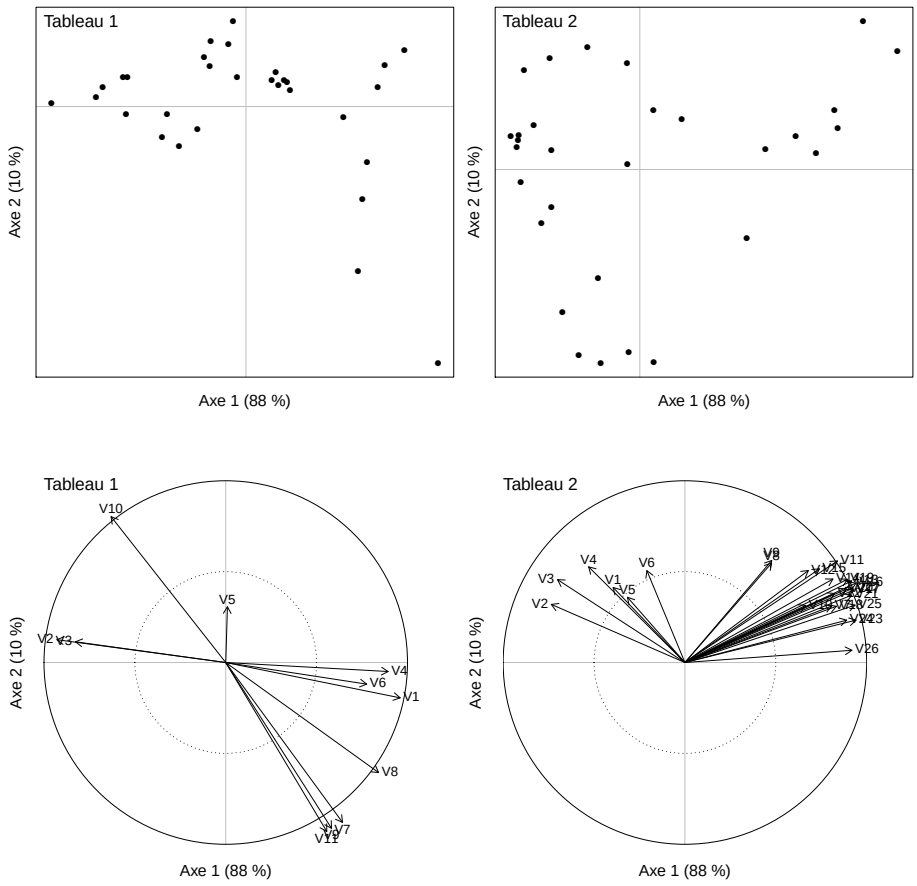
Pour les variables quantitatives, l'interprétation est basée sur la comparaison des deux cercles des corrélations : deux flèches pointant dans la même direction ou dans des directions diamétralement

opposées indiquent deux variables corrélées (voir fiche **89**). Le sens de la corrélation est indiqué par le *signe* de la corrélation entre les axes d'une même paire. Ces corrélations entre axes sont données par la fonction `MVA.synt()`[1], dans le second tableau qu'elle renvoie. Si la corrélation pour une paire d'axes est *positive*, les variables dirigées vers la *gauche* de cet axe pour le premier tableau sont positivement reliées aux variables dirigées vers la *gauche* du même axe pour le second tabeau. Si la corrélation pour la paire d'axes est *négative*, les variables dirigées vers la *gauche* de cet axe pour le premier tableau sont positivement reliées aux variables dirigées vers la *droite* du même axe pour le second tabeau.

Pour les variables qualitatives l'interprétation est similaire à l'ACM : plus deux modalités sont proches et plus elles sont associées (voir fiche **99**). Là encore, le sens de l'association dépend de la corrélation entre paires d'axes.

Les relations entre variables quantitatives et qualitatives s'interprètent comme en analyse mixte : plus le barycentre d'une modalité est situé vers l'extrémité d'une flèche (quand on le projette perpendiculairement à cette flèche) et plus la valeur moyenne des individus de cette modalité est extrême pour la variable quantitative représentée par la fèche (le sens de la relation étant donné par la corrélation entre paires d'axes).

Remarque : les relations entre variables d'un même tableau s'interprètent comme en ACP (voir fiche **97**), ACM (voir fiche **99**) et analyse mixte (voir fiche **100**).



115. L'analyse de de co-inertie procustéenne

— PACKAGE(S) ADDITIONNEL(S) UTILISÉ(S) DANS CETTE FICHE —

[1]`ade4`, [2]`RVAideMemoire`

SYNONYMES, TRADUCTIONS ET ABRÉVIATIONS

Français – Analyse de co-inertie procustéenne
Anglais – *Procustean co-inertia analysis* (*PCIA*)

Préparation des données

La PCIA généralise l'analyse procustéenne (voir fiche **113**) au cas où au moins un tableau contient plus de deux colonnes. Les deux tableaux peuvent être des tableaux de variables, ou des axes d'une précédente ordination (voir fiche **90** pour récupérer les coordonnées des individus sur ces axes). Toutes les variables doivent être quantitatives, et une standardisation de ces variables est souvent bénéfique (voir fiche **88**).

Réalisation de l'analyse

Pour réaliser la PCIA : `PCIA<-procuste(tableau1,tableau2)`[1] où `tableau1` et `tableau2` sont les deux tableaux. Le premier est appelé X et le second Y.

Qualité de l'analyse

On peut à la fois estimer la « concordance » entre les deux tableaux et la tester *via* `randtest(PCIA)`[1]. La fonction renvoie une statistique appelée m^2 (`Observation`) qui varie entre 0 (concordance parfaite) et 1 (aucune concordance), et le résultat d'un test non paramétrique appelé PROTEST (basé sur le m^2). Une *p-value* significative indique qu'il y a une concordance significative entre les deux tableaux.

Représentations graphiques

En PCIA on a deux représentations possibles : le *graphe des individus*, qui montre la position des individus à la fois pour le tableau X et le tableau Y sur un *plan factoriel* composé de deux axes, et le *cercle des corrélations* (voir fiche **89**) qui permet d'interpréter la répartition des individus.

Remarque : en PCIA chaque individu a deux coordonnées dans le même plan factoriel, l'une pour le tableau X et l'autre pour le tableau Y. Il y a donc deux nuages de points dans le même espace.

Graphe des individus

Pour tracer le graphe : `MVA.plot(PCIA,"pairs")`[2]. Chaque individu est représenté par une flèche. Celle-ci démarre aux coordonnées de l'individu pour le tableau X, et s'arrête aux coordonnées de ce même individu pour le tableau Y.

Voir `?MVA.pairplot` pour bien d'autres options graphiques.

Cercle des corrélations

Pour tracer le graphe : `MVA.plot(PCIA,"corr")`[2]. Par défaut les variables des deux tableaux sont représentées. Pour ne représenter que les variables du tableau X ajouter l'argument `set=1`, pour les variables du tableau Y `set=2`.

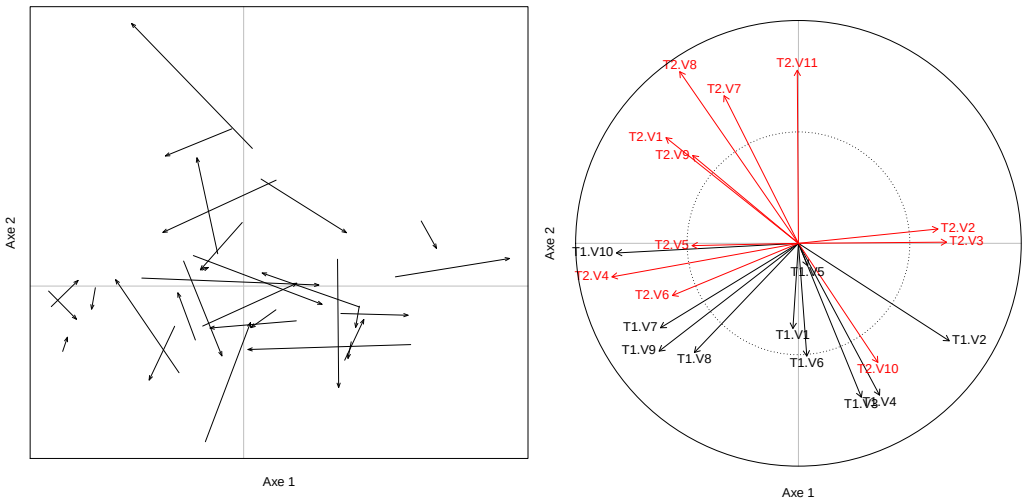
Pour supprimer les flèches, ajouter l'argument `arrows=FALSE`. Pour représenter différemment les variables des différents tableaux, ajouter l'argument `fac=facteur` où `facteur` est le facteur définissant le groupe (*i.e.* le tableau) de chaque variable. Les arguments `col`, `pch` et `lwd` permettent de personnaliser une telle représentation. Voir `?MVA.corplot` pour bien d'autres d'options graphiques.

Interprétation

Plus les flèches du graphe des individus sont courtes et plus cela indique une concordance forte entre les deux tableaux (*i.e.* les individus sont à peu près positionnés au même endroit quel que soit le tableau). À l'inverse, plus les flèches sont longues et partent dans tous les sens, et plus cela indique une concordance faible entre les deux tableaux. Il y aura toujours une cohérence entre le graphe et le résultat du test statistique.

Remarque : la longueur des flèches n'est jamais identique pour tous les individus ; on peut ainsi identifier lesquels sont « semblables » dans les deux tableaux ou au contraire très différents.

À la condition que la concordance entre les deux tableaux soit significative, le cercle des corrélations permet d'identifier les variables qui sont associées entre ces tableaux. Son interprétation est identique à un cercle des corrélations classique (voir fiche **89**).



116. L'analyse canonique des corrélations régularisée généralisée

PACKAGE(S) ADDITIONNEL(S) UTILISÉ(S) DANS CETTE FICHE

[1]`RVAideMemoire`, [2]`mixOmics`

SYNONYMES, TRADUCTIONS ET ABRÉVIATIONS

Français – Analyse canonique des corrélations régularisée généralisée
Anglais – *Regularized generalized canonical correlation analysis* (*RGCCA*)

Préparation des données

La RGCCA exige que toutes les variables soient quantitatives. Ce peuvent être des variables mesurées telles quelles et/ou des axes d'une précédente ordination (voir fiche **90** pour récupérer les coordonnées des individus dans ce dernier cas). Les différents tableaux (appelés « blocs ») n'ont pas à contenir le même nombre de variables.

L'analyse fonctionne d'autant mieux que les variables ont une distribution à peu près normale (au moins symétrique) et que dans chaque tableau elles sont reliées entre elles par des relations linéaires. Une transformation préalable des données peut grandement aider à améliorer la situation (voir fiche **88**). Une standardisation des variables de chaque tableau est également souvent bénéfique, car elle permet de donner le même poids à toutes les variables (voir fiche **88**).

Remarque : on peut intégrer un facteur dans l'analyse en le transformant en *variables indicatrices*, *via* `var.ind<-dummy(facteur)`[1] où `facteur` est un vecteur définissant le groupe de chaque individu. Le tableau `var.ind` est ensuite traité comme les autres, excepté pour la transformation qui n'est jamais nécessaire. Cette façon de procéder permet d'utiliser la RGCCA comme analyse discriminante.

Tous les tableaux doivent être rassemblés dans une liste. Pour cela : `blocs<-list(bloc1=tableau1, bloc2=tableau2...)` où `tableau1`, `tableau 2` et `...` sont les différents tableaux (qui seront nommés « bloc1 », « bloc2 »… dans l'analyse). Les compartiments de la liste doivent obligatoirement avoir un nom.

Pour finir, il est nécessaire de définir les relations interblocs à analyser (*i.e.* les tableaux à relier). Rien n'oblige en effet à ce que l'analyse cherche l'information commune à toutes les paires de tableaux deux-à-deux. Au contraire, il vaut mieux définir les relations qui sont les plus pertinentes vis-à-vis des questions biologiques. Ces relations sont définies dans une matrice symétrique (*design matrix*) à autant de lignes et de colonnes que de blocs, remplie de 0 (pas de relation) et de 1 (relation). La diagonale est nécessairement constituée uniquement de 0.

EXEMPLE(S)

Les blocs 1 et 2 sont à relier chacun au bloc 3, mais pas entre eux :

```
> matrice <- matrix(c(0,0,1,0,0,1,1,1,0),nrow=3)
> matrice
     [,1] [,2] [,3]
[1,]    0    0    1
[2,]    0    0    1
[3,]    1    1    0
```

Réalisation de l'analyse

Pour réaliser la RGCCA : `RGCCA<-wrapper.rgcca(blocs,C=matrice,tau="optimal",ncomp=nb)`[2] où `matrice` est la *design matrix* et `nb` le nombre d'axes à construire.

Remarque 1 : si l'argument `C` n'est pas fourni toutes les paires de tableaux deux-à-deux sont considérées.

Remarque 2 : par défaut la fonction standardise tous les jeux de données, ce qui est souvent intéressant car cela permet de donner le même poids à toutes les variables (voir fiche **88**). Pour ne pas standardiser, ajouter l'argument `scale=FALSE`.

Remarque 3 : la RGCCA créé autant d'ordinations séparées qu'il y a de tableaux. On a donc un axe 1 pour chaque bloc, un axe 2 pour chaque bloc et ainsi de suite. Si l'argument `ncomp` n'est pas précisé un seul axe est construit par bloc.

Remarque 4 : la RGCCA a pour objectif de maximiser une certaine information qui se situe sur un continuum à deux extrêmes : (i) la *variance totale intrabloc* expliquée et (ii) la *corrélation* entre les axes de même niveau (*i.e.* les axes 1, les axes 2...). L'argument `tau` permet de préciser où l'on doit se situer sur ce continuum. On peut ainsi choisir de maximiser plutôt l'information de chaque bloc expliquée par l'analyse (extrême « variance », *via* `tau=rep(1,nb)` où `nb` est le nombre de blocs), plutôt la corrélation entre les axes de même niveau (extrême « corrélation », *via* `tau=rep(0,nb)`) ou de trouver le meilleur compromis entre ces deux extrêmes (`tau="optimal"`, intéressant par défaut).

Qualité de l'analyse

On estime la qualité de l'analyse par le pourcentage de variance totale intrabloc expliqué par chaque axe (de chaque bloc), et par la corrélation entre les axes qui doivent être reliés selon la *design matrix*. Toutes ces valeurs sont obtenues *via* `MVA.synt(RGCCA)`[1], dans les différents tableaux renvoyés par la fonction (un par critère et par bloc).

Remarque : contrairement à la plupart des analyses multivariées, les pourcentages de variance expliqués ne sont pas nécessairement en ordre décroissant (même si c'est le cas le plus souvent). Les corrélations entre axes sont elles toujours de moins en moins fortes.

Représentations graphiques

En RGCCA on a deux représentations possibles *pour chaque bloc* : le *graphe des individus* qui montre la position de ces individus sur un *plan factoriel* composé de deux axes, et le *cercle des corrélations* (voir fiche **89**) qui permet d'interpréter la répartition des individus.

Graphe des individus

Pour tracer le graphe du premier bloc : `MVA.plot(RGCCA,space=1)`[1]. Pour le bloc suivant : `MVA.plot(RGCCA,space=2)`[1]. Et ainsi de suite.

Les axes 1 (horizontal) et 2 (vertical) sont représentés par défaut, ils peuvent être changés grâce aux arguments `xax` et `yax`. Voir `?MVA.scoreplot` pour bien d'autres d'options graphiques.

Cercle des corrélations

Pour tracer le graphe du premier bloc : `MVA.plot(RGCCA,"corr",space=1)`[1]. Pour le bloc suivant : `MVA.plot(RGCCA,"corr",space=2)`[1]. Et ainsi de suite.

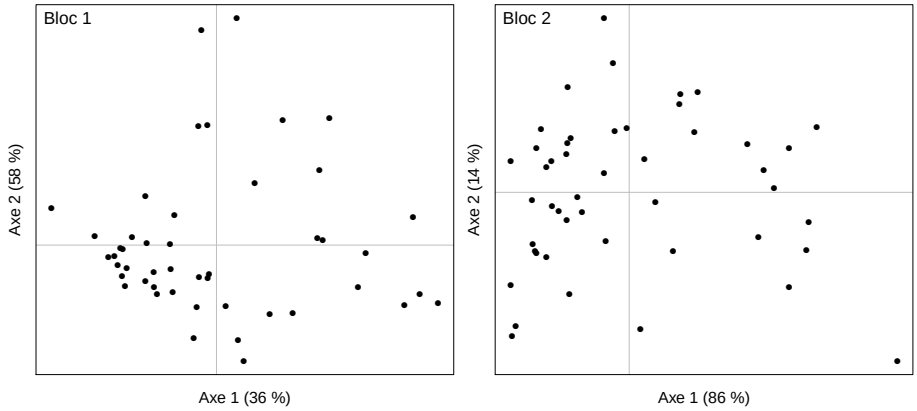
Pour supprimer les flèches, ajouter l'argument `arrows=FALSE`. Voir `?MVA.corplot` pour bien d'autres d'options graphiques.

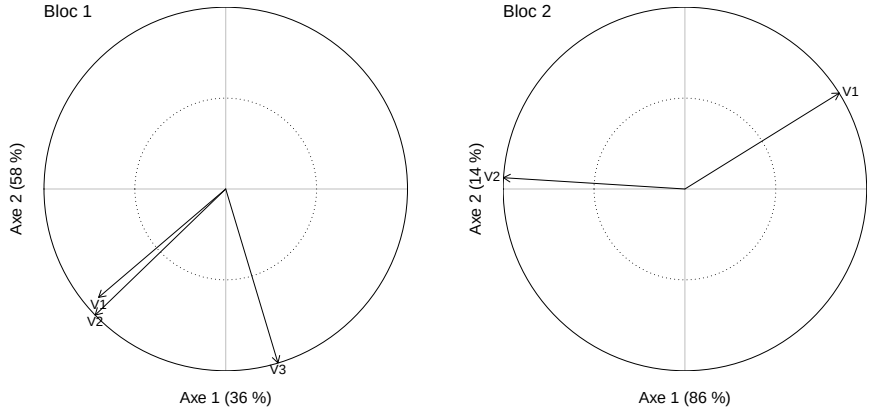
Interprétation

L'objectif de la RGCCA est le plus souvent d'identifier les variables de chaque tableau qui sont associées entre elles, mais aussi (et surtout) les associations entre variables de tableaux différents. À ce titre l'interprétation se base essentiellement sur les cercles des corrélations.

Les corrélations interblocs (*i.e.* entre axes de même niveau) indiquent sur quels axes se concentrer pour l'interprétation. On ne retient que les axes très corrélés entre eux, car c'est cette corrélation (*i.e.* cette information commune à plusieurs tableaux) qui est la cible de l'analyse et donc la base de l'interprétation. Si par exemple la corrélation est forte pour les axes 1, on repère pour chaque tableau les variables les plus corrélées à ce premier axe (voir fiche **89**). Ce sont ces variables qui sont les plus associées.

Au sein d'un tableau, l'interprétation est simple : les variables aux extrémités opposées d'un axe sont corrélées négativement (voir fiche **89**). Pour la relation entre variables de deux tableaux différents, si la corrélation pour la première paire d'axes est *positive*, les variables dirigées vers la *gauche* de l'axe 1 pour le premier tableau sont positivement reliées aux variables dirigées vers la *gauche* de l'axe 1 pour le second tabeau. Si la corrélation pour la première paire d'axes est *négative*, les variables dirigées vers la *gauche* de l'axe 1 pour le premier tableau sont positivement reliées aux variables dirigées vers la *droite* de l'axe 1 pour le second tabeau.


Bloc 1
Axe 2 (58 %)
Axe 1 (36 %)
Bloc 2
Axe 2 (14 %)
Axe 1 (86 %)


Bloc 1
Axe 2 (58 %)
V1
V2
V3
Axe 1 (36 %)
Bloc 2
Axe 2 (14 %)
V1
V2
Axe 1 (86 %)

117. L'analyse procustéenne généralisée

— PACKAGE(S) ADDITIONNEL(S) UTILISÉ(S) DANS CETTE FICHE —

[1]`FactoMineR`, [2]`RVAideMemoire`

SYNONYMES, TRADUCTIONS ET ABRÉVIATIONS

Français – Analyse procustéenne généralisée
Anglais – *Generalized Procrustes analysis* (*GPA*)
Generalized Procrustes superimposition

Préparation des données

La GPA exige que toutes les variables soient quantitatives. Ce peuvent être des variables mesurées telles quelles et/ou des axes d'une précédente ordination (voir fiche **90** pour récupérer les coordonnées des individus dans ce dernier cas). Les différents tableaux n'ont pas à contenir le même nombre de variables.

L'analyse fonctionne d'autant mieux que les variables ont une distribution à peu près normale (au moins symétrique) et que dans chaque tableau elles sont reliées entre elles par des relations linéaires. Une transformation préalable des données peut grandement aider à améliorer la situation (voir fiche **88**).

Pour finir, tous les tableaux doivent être rassemblés dans un unique tableau. Pour cela : `tableau<-data.frame(tableau1,tableau2...)` où `tableau1`, `tableau 2` et `...` sont les différents tableaux (de gauche à droite dans le tableau final).

Réalisation de l'analyse

Pour réaliser la GPA : `GPA<-GPA(tableau,group=colonnes,graph=FALSE)`[1] où `colonnes` est un vecteur donnant le nombre de colonnes de chaque tableau (de la gauche vers la droite), afin que la fonction puisse les séparer en interne. Pour donner un nom à chaque tableau (ce qui peut faciliter l'interprétation graphique ultérieure), ajouter l'argument `name.group=noms` où `noms` est un vecteur contenant le nom de chaque tableau (de la gauche vers la droite). Par défaut ils sont nommés `group.1`, `group.2`...

Remarque : par défaut la fonction standardise tous les jeux de données, ce qui est souvent intéressant car cela permet de donner le même poids à toutes les variables (voir fiche **88**). Pour ne pas standardiser, ajouter l'argument `scale=FALSE`.

Qualité de l'analyse

Test

L'objectif de la GPA est de trouver une *configuration consensus*, *i.e.* une « moyenne » des ordinations respectives de chaque tableau, après que celles-ci aient subi les étapes de translation, rotation et homothétie qui les ajustent au mieux les unes par rapport aux autres. C'est souvent cette configuration consensus qui est l'objectif de l'analyse, afin d'être interprétée ou utilisée dans une analyse ultérieure (voir fiche **90** pour en récupérer les coordonnées).

Remarque : la première étape de la GPA est en fait de réaliser une ACP sur chaque tableau, d'où l'obligation que les variables soient toutes quantitatives (voir fiche **97**). Ce sont ces ACP qui sont ensuite ajustées les unes sur les autres.

On peut tester si la configuration consensus est significative, *i.e.* s'il y a réellement concordance entre les différents tableaux. Pour réaliser le test : `GPA.test(tableau,group=colonnes)`[2]. Le temps de calcul nécessaire pour ce test est relativement long.

Synthèse

Si la concordance entre les différents tableaux est significative, un certain nombre d'indicateurs permettent d'estimer la qualité de l'analyse. La manière dont la GPA identifie la configuration consensus est de capter un maximum de la *variance totale* de l'ensemble des tableaux. On s'intéresse donc au

pourcentage de variance totale globalement capté, et plus précisément pour chaque axe de cette ordination consensus. Ces pourcentages sont obtenus *via* `MVA.synt(GPA)`[2], dans les deux premiers tableaux renvoyés par la fonction. Le troisième tableau renvoyé diffère du deuxième car il donne le pourcentage de *variance consensus* expliqué par chaque axe. Traditionnellement on utilise plutôt les pourcentages de variance totale.

Une autre manière d'estimer la qualité de l'analyse est d'identifier si tous les tableaux sont aussi bien représentés par la configuration consensus. Pour cela on examine la *variance résiduelle, i.e.* celle qui n'est pas captée par la configuration consensus. Si la synthèse est efficace pour tous les tableaux, cette variance résiduelle devrait se répartir équitablement entre eux. Si un ou quelques tableaux ont un pourcentage de variance résiduelle bien plus élevé que les autres tableaux, c'est que la configuration consensus n'est pas à même de bien les représenter (cependant il est important de prendre en compte le pourcentage de variance consensus global, car si celui-ci est très élevé on peut considérer que les tableaux sont tous plutôt bien représentés, même si certains le sont moins bien que d'autres). La répartition de la variance résiduelle est fournie dans le dernier tableau renvoyé par la fonction `MVA.synt()`[2].

Représentations graphiques

En GPA on a deux représentations graphiques possibles : le *graphe des individus* qui montre la position de ces individus sur un *plan factoriel* composé de deux axes, et éventuellement le *cercle des corrélations* (voir fiche **89**) si l'interprétation en termes de variables originales est l'objectif de l'analyse.

Graphe des individus

Pour tracer le graphe : `MVA.plot(GPA)`[2]. C'est la configuration consensus qui est représentée.

Les axes 1 (horizontal) et 2 (vertical) sont représentés par défaut, ils peuvent être changés grâce aux arguments `xax` et `yax`. Pour ajouter des groupes sur le graphe, utiliser l'argument `fac=facteur` où `facteur` est le facteur définissant le groupe de chaque individu. Les arguments `col`, `pch`, `fac.lab`, `contours`, `stars` et `barycenters` permettent de personnaliser une telle représentation. Voir `?MVA.scoreplot` pour bien d'autres d'options graphiques.

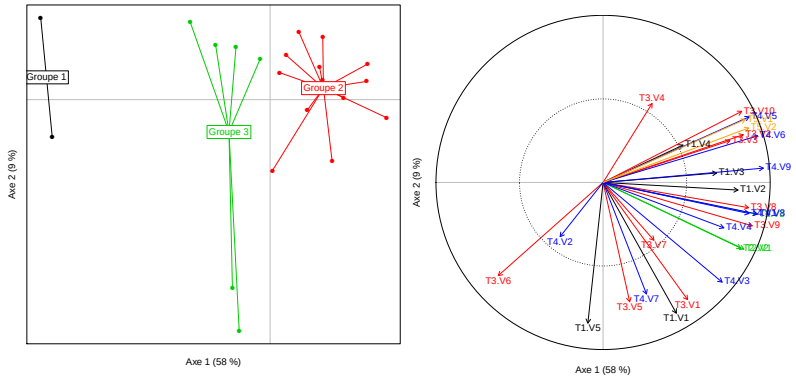
Pour représenter les individus dans toutes les ordinations : `plot(GPA)`. Les points d'un même individu sont reliés au point de l'individu dans la configuration consensus. On peut ainsi identifier les individus qui sont « semblables » dans les différents tableaux, ou au contraire très différents (donc pour lesquels le consensus est probablement moins pertinent).

Cercle des corrélations

Pour tracer le graphe : `MVA.plot(GPA,"corr")`[2]. Pour supprimer les flèches, ajouter l'argument `arrows=FALSE`. Pour représenter différemment les variables des différents tableaux, ajouter l'argument `fac=facteur` où `facteur` est le facteur définissant le groupe (*i.e.* le tableau) de chaque variable. Les arguments `col`, `pch` et `lwd` permettent de personnaliser une telle représentation. Voir `?MVA.corplot` pour bien d'autres d'options graphiques.

Interprétation

Si l'interprétation de la configuration consensus est l'objectif de l'analyse, celle-ci est identique à l'interprétation d'une ACP (voir fiche **97**).



Bibliographie et ouvrages/documents/liens recommandés

Livres

Crawley M.J. (2007) The R Book. John Wiley & Sons, inc.

Dagnelie P. (2006a) Statistique théorique et appliquée. 1. Statistique descriptive et base de l'inférence statistique. 3ème édition. De Boeck.

Dagnelie P. (2006b) Statistique théorique et appliquée. 2. Inférence statistique à une et à deux dimensions. 3ème édition. De Boeck.

Legendre P. and **Legendre L.** (2012) Numerical ecology. 3rd edition. Elsevier.

Borcard D., **Gillet F.** and **Legendre P.** (2012) Numerical ecology with R. Springer.

Millot G. (2008) Comprendre et réaliser les tests statistiques à l'aide de R. 2ème édition. De Boeck.

Documents en ligne

Bates D. (2010) lme4 :Mixed-effects modeling with R. [*http://lme4.r-forge.r-project.org*]

Champely S. (2005) Introduction à l'analyse multivariée (factorielle) sous R. [*http://pierre.jeandenand.free.fr/Cours/Statistiques%20avec%20Rgui.pdf*]

Dagnelie P. (2003) Principes d'expérimentation : planification des expériences et analyse de leurs résultats. Les presses agronomiques de Gembloux. [*http://www.dagnelie.be*]

Fox J. (2002) Cox proportional-hazards regression for survival data. [*http://cran.r-project.org*, rubrique *Contributed* : *Web Appendix to the book "An R and S-PLUS Companion to Applied Regression"*]

Genolini C. (2002) R, bonnes pratiques. [*http://cran.r-project.org*, rubrique *Contributed*]

Paradis E. (2002) R pour les débutants. [*http://cran.r-project.org*, rubrique *Contributed*]

Poinsot D. (2004) Statistiques pour statophobes. [*http://perso.univ-rennes1.fr/denis.poinsot*]

Poinsot D. (2005) R pour les statophobes. [*http://perso.univ-rennes1.fr/denis.poinsot*]

Les nombreux cours en ligne de **D. Chessel**, **A.B. Dufour**, **J.R. Lobry** et **M. Royer** : *http://pbil.univ-lyon1.fr/R/enseignement.html*

Les cours en ligne de **M-L. Delignette-Muller** : *http://www2.vet-lyon.fr/ens/biostat/accueil.html*

Un site très complet et très bien fait sur les analyses multivariées en écologie des communautés : *https://sites.google.com/site/mb3gustame*

Le site de R : *http://www.R-project.org*.

Groupes d'utilisateurs

Le forum du groupe des utilisateurs du logiciel R : *http://forums.cirad.fr/logiciel-R/index.php*

Semin-R, un autre groupe d'utilisateurs de R : *http://rug.mnhn.fr/semin-r*