Introduction à l'environnement `R` pour l'analyse de données quantitatives en psychologie

Enseignement de Master de Psychologie

Jean-Luc Kop
Université de Lorraine

2018-01-02

Contents

Préambule

Ce document est un support de cours destiné à accompagner l'enseignement de méthodologie d'analyse quantitative de Master de Psychologie de l'Université de Lorraine (site de Nancy). Il n'a nullement la prétention de remplacer les nombreuses introductions au logiciel `R` (R Core Team, 2017b) qui sont déjà disponibles sur Internet et qui sont généralement plus exhaustives et détaillées que ne l'est ce document (voir, par exemple, Online R resources for Beginners).

Même s'il est possible avec un peu d'attention et de persévérance d'utiliser ce document en auto-formation, les explications données au fil de l'enseignement sont quasi indispensables pour permettre à des néophytes de bien comprendre certaines subtilités. Il en est de même des exercices qui seront distribués et commentés au fil des travaux dirigés.

La maîtrise globale du logiciel `R` n'est évidemment pas l'objectif de ce document ni de l'enseignement qui l'accompagne. Il s'agit plus modestement de permettre aux futurs utilisateurs d'acquérir suffisamment de connaissances et d'autonomie pour mener à bien les analyses statistiques dont ils ont et auront besoin dans le cadre de leurs recherches fondamentales ou plus appliquées.

Les remarques, critiques et suggestions sur ce document sont à adresser à `jean-luc.kop [at] univ-lorraine.fr`, à qui les lecteurs pourront aussi signaler les erreurs qui peuvent subsister.

Conception du document

Ce document a été réalisé grâce aux paquets `bookdown` (Xie, 2016) et `knitr` (Xie, 2014) qui s'appuient sur R Mardown implémenté dans le logiciel R Studio. Il existe en deux formats :

- un format html (style *GitBook*)
- un format LaTeX/PDF

Licence

Chapter 1

Introduction

Plus qu'un logiciel, `R` est à la fois un puissant langage de programmation et un outil permettant de réaliser des analyses statistiques et des représentations graphiques. Il a été créé par deux enseignants-chercheurs néo-zélandais (Ross Ihaka et Robert Gentleman) de l'Université d'Auckland au début des années 1990, en s'inspirant notamment d'un langage pour les statistiques appelé `S`. Le nom `R` finalement choisi est en référence à cet autre langage, ainsi qu'un petit clin d'oeil aux initiales des prénoms de ses deux créateurs. La version 1.0.0 de `R` est publiée en 2000. Depuis, il ne cesse de se développer, grâce à la contribution de nombreux informaticiens et statisticiens bénévoles dans le monde entier et il compte actuellement plusieurs millons d'utilisateurs.

L'environnement `R` se développe autour d'un noyau central dont le programme est mis à jour plusieurs fois par an et qui comprend les fonctionnalités les plus courantes. Il s'enrichit d'une multitude[1] de paquets (*packages*) spécialisés écrits par des utilisateurs en fonction de besoins spécifiques. Ces paquets sont un ensemble de fonctions qui ont la particularité d'être écrits le plus souvent directement dans le langage R et n'importe quel utilisateur un peu averti peut assez rapidement écrire ses propres fonctions. Lorsqu'un paquet est suffisamment abouti (correction des bugs, écriture de la documentation associée), il est mis à la disposition de la communauté. La liste des packages disponibles, leur description et les instructions pour leur installation sont disponibles ici.

Dans sa version de base, l'environnement graphique de `R` est assez fruste (figure 1.1): le menu ne comporte que peu de fonctions, la connaissance du langage `R` est nécessaire et on ne peut écrire qu'une ligne à la fois. Pour rendre l'environnement plus convivial, plusieurs interfaces graphiques (GUI) existent (`DeduceR` (Fellows (2012)), `Rattle` (Williams (2011)), `RKWard` (Michalke (2014)), `JGR` (Helbig, Theus, & Urbanek (2005)), ...), disponibles sous forme de paquets. La plus connue de ces interfaces est sans doute celle créée par John Fox qui s'appelle `RCommander` (Fox (2017)).

[1] Le seuil des 10000 paquets officiels a été franchi en 2017

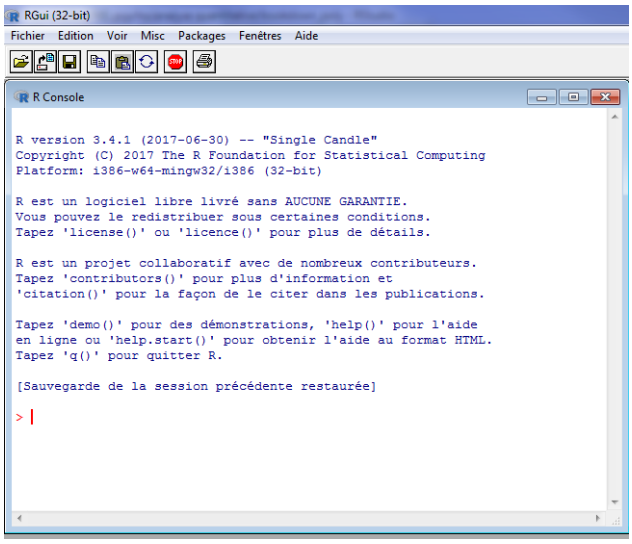


Figure 1.1: La console du logiciel R

`R` est notamment renommé pour l'excellente qualité des graphiques qu'il permet de réaliser (figure 1.2). On trouvera de nombreux exemple sur ce site. Actuellement, le paquet `ggplot2` (Wickham (2009)) s'impose comme étant la principale référence pour la réalisation de graphiques d'un très grand esthétisme (voir ici, par exemple).

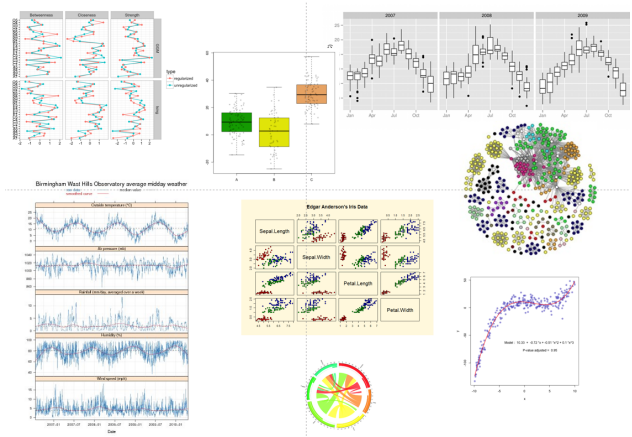


Figure 1.2: Exemples de graphiques avec R

Au-delà de ces graphiques *statiques*, il est même possible de réaliser avec `R` :

- des graphiques animés (voir ici pour un exemple) avec les paquets `animation` (Xie (2013)), `anim.plots` (Hugh-Jones (2017)), `tweenr` (Pedersen (2016)), ... ;
- des graphiques interactifs avec le paquet `Plotly` (Sievert et al. (2017)) ou `ggvis` (Chang & Wickham (2016)) (Cliquez sur les liens pour des exemples) ;
- ou même des applications interactives déployées sur le web avec `Shiny` (Chang, Cheng, Allaire, Xie, & McPherson (2017)) (voir ici pour de nombreux exemples).

La documentation sur `R` est pléthorique et facilement accessible, que ce soit sous forme d'ouvrages imprimés ou d'innombrables documents sur internet[2]. Les sites consacrés à `R` ou à certains de ces paquets sont beaucoup

[2]La combinaison de mots clés comme "R" et "tutorial" ou "manuals" ou "documentation" dans un moteur de recherche

trop nombreux ajourd'hui pour en trouver une recension exhaustive. A simple titre d'exemple, on peut citer, pour les francophones, le site collaboratif Abcd'R. Un autre site très utile est le moteur de recherche RDocumentation qui permet de chercher (et de trouver !) très rapidement des mots clés dans n'importe quel paquet de `R` (y compris les paquets en développement qui ne sont pas encore publiés).

Il est difficile d'extraire seulement quelques références parmi toute cette documentation. On pourra tout de même conseiller, plus particulièrement pour les psychologues :

- un site de ressources en ligne pour les débutants (ouvrages, vidéos, sites web, ...) : introductoryR
- d'autres ressources compilées sur le blog de Jeromy Anglim
- l'ouvrage de Daniel Navarro : *Learning statistics with R: A tutorial for psychology students and other beginners* (ou ici pour un autre lien)
- le site de William Revelle qui fournit de nombreux tutoriels et un paquet `psych` qui sera très utilisé dans ce document :
- L'ouvrage de Jonathan Baron : *Notes on the use of R for psychology experiments and questionnaires*

permettra de s'en convaincre facilement

Première partie : Présentation générale de R et de son langage

Chapter 2

Installation de l'environnement R

2.1 Le logiciel R de base

Sur le site internet de R (https://cran.r-project.org/), il suffit de sélectionner le téléchargement du logiciel correspondant à son système d'exploitation : Windows, Macintosh ou Linux. La suite dépend du système d'exploitation, mais ne pose pas de problème particulier. Par exemple, sous Windows, il faut :

- dans la nouvelle page qui s'ouvre, suivre le lien `base` ;
- cliquer sur le lien `Download R 3.4.1 for Windows`[1] ;
- exécuter le fichier téléchargé (ce qui lancera le programme d'installation) ;
- Validez les options proposées par défaut.

2.2 Installation des paquets

L'installation du logiciel R de base peut s'enrichir par l'installation de paquets. Pour cela il faut lancer le programme R (ce qui se fait comme pour n'importe quel autre programme). On obtient alors une fenêtre semblable à celle de la figure 1.1.

2.2.1 Installation automatique

L'installation automatique est la plus simple. Elle nécessite une connexion internet active.

Il suffit d'utiliser le menu `packages` puis "*installer le(s) package(s)*". Les paquets sont hébergés dans différents sites miroirs répartis dans le monde : il est préférable de choisir un site proche géographiquement[2]. La dernière étape consiste à choisir le paquet à installer dans la longue liste (classée par ordre alphabétique) qui apparaît alors à l'écran. Le paquet `psych` peut être installé à titre d'exemple, car il sera utilisé fréquemment dans la suite de ce document.

2.2.2 Installation manuelle

On peut télécharger les paquets manuellement et les installer ensuite dans R. Pour le système d'exploitation windows, voici les différentes étapes :

[1] Le numéro de la version change au fur et à mesure des mises à jour.
[2] Il peut arriver qu'un site ne fonctionne pas : il suffit alors d'en choisir un autre.

- sur la page d'accueil du site R (https://cran.r-project.org/), suivre le lien Packages dans la rubrique "*Software*" à gauche de la page ;
- cliquer sur le lien "*Table of available packages, sorted by name*" ;
- cliquer sur le lien correspondant au paquet souhaité ;
- télécharger le fichier "*zip*" dans la rubrique "*Windows binaries*", "*r-release*" ;
- Lancer R puis, dans le menu packages, sélectionner "*install package(s) from local files*".

2.3 Chargement des paquets

Il est important de distinguer l'*installation* des paquets de leur *chargement*. Une fois un package installé, il est disponible sur l'ordinateur où R est installé. Mais pour que les commandes de ce paquet soient accessibles dans R, il faut le **charger**. Voila comment procéder :

- dans R, menu packages, sélectionner "charger le package"
- choisir le paquet souhaité dans la liste proposée[3].

Le chargement d'un paquet est annulé à chaque fois que l'on quitte le programme R. L'installation est permanente, alors que le chargement est temporaire.

> Certains paquets nécessitent l'installation et le chargement d'autres paquets Avec l'installation automatique par internet, les paquets subordonnés sont installés automatiquement. Ce n'est pas forcément le cas avec une installation manuelle (il faut alors les installer un à un).

2.4 L'environnement RStudio

On l'a vu (figure 1.1), l'environnement R de base est très fruste. Le logiciel RStudio est venu remédier à cette faiblesse en offrant un environnement beaucoup plus ergonomique. RStudio est un IDE (*Integrated Development Environment*) pour R qui comprend de nombreux outils très efficaces (liste non exhaustive) :

- un éditeur de texte performant pour l'écriture des commandes (avec notamment une complétion automatique du langage) ;
- des outils d'aide ;
- un gestionnaire des objets créés dans R et des paquets ;
- la possibilité de créer facilement des documents mis en forme comme celui que vous êtes en train de lire (R Markdown) ;
- la possibilité de créer des applications internet interactives (Shiny).

Tout comme R, RStudio est un logiciel gratuit[4] disponible pour les trois principaux systèmes d'exploitation : windows, macintosh et linux. La procédure pour installer RStudio est simple[5] :

- se connecter à : https://www.rstudio.com/products/RStudio/ ;
- choisir l'installation sur ordinateur de bureau (desktop)
- puis l'édition "open source" (free)gratuite] ;
- et enfin puis la version correspondant à votre système d'exploitation).

Lorsque RStudio est installé, son exécution appelle automatiquement celle de R. L'environnement RStudio divise schématiquement l'écran en 4 fenêtres (cf. Figure 2.1) correspondant à :

[3] Seuls les paquets installés apparaissent dans la liste.

[4] Dans sa version "*open source*" ; il existe aussi une version commerciale à destination des entreprises qui offre des fonctionnalités et des services supplémentaires.

[5] le logiciel R soit avoir été installé au préalable

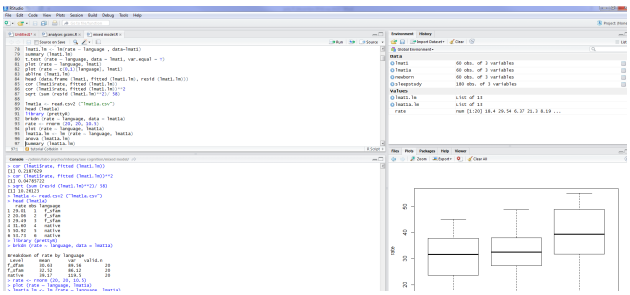


Figure 2.1: L'environnement RStudio

- fenêtre en haut à gauche : éditeur des commandes du langage R (*script*) ;
- fenêtre en bas à gauche : c'est la fenêtre originale de R (la *console*) ; elle contient notamment les résultats des commandes ;
- fenêtre en haut à droite ; elle contient par défaut la liste des différents objets créés pendant la session ; on peut aussi y afficher l'historique de toutes les commandes utilisées ;
- fenêtre en bas à droite ; selon le contexte, on y trouvera un explorateur de fichiers, les graphiques réalisés, un gestionnaire des packages ou les fichiers d'aide.

Chapter 3

Les fondamentaux

3.1 L'environnement de base de R

Comme indiqué déjà plusieurs fois, l'environnement de base de R est assez fruste et se présente sous forme d'une fenêtre avec un menu qui ne comporte que quelques fonctions de base pour gérer l'environnement et une console qui reçoit les commandes et qui affiche les résultats (Figure 3.1).

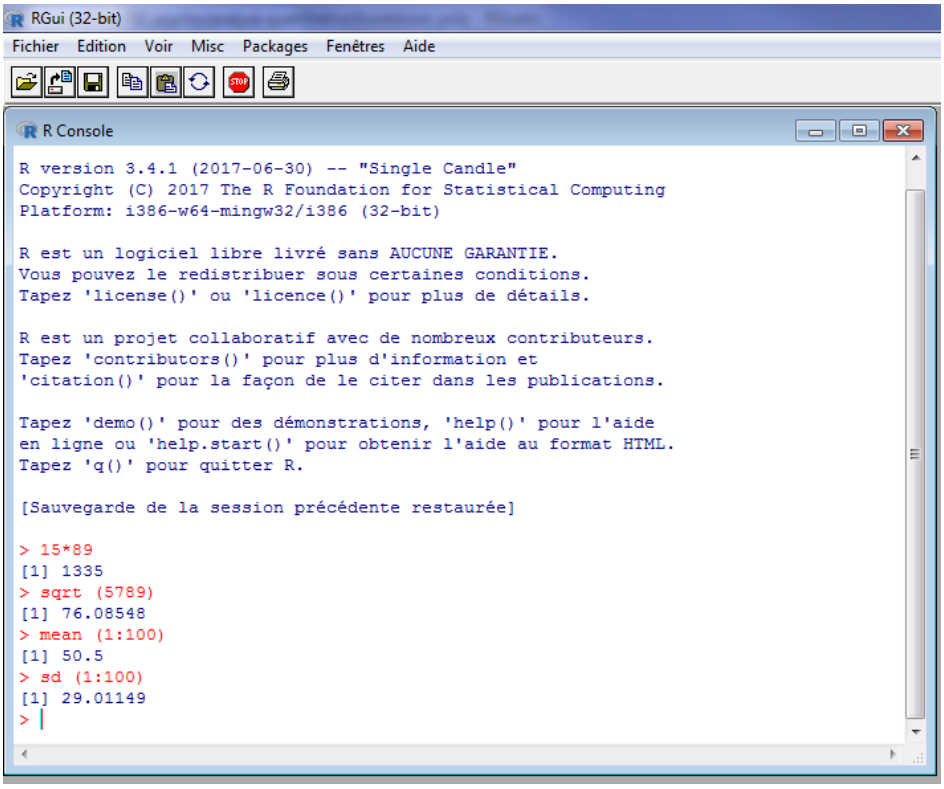


Figure 3.1: La console R avec quelques commandes et résultats

Le langage R s'appuie sur des fonctions selon la syntaxe générale suivante :

```
fonction (objet, option1, option2,... )
```

Le nom de la fonction est suivi, entre parenthèses, d'un ou de plusieurs arguments (l'objet sur lequel va porter la commande et des options) de la fonction. Même lorsqu'il n'y a pas d'arguments, les parenthèses sont nécessaires[1]. Voici quelques exemples :

- `mean (c(10,12,14,15,18,10))` permet d'obtenir la moyenne de l'ensemble des valeurs indiquées ;
- `help.start()` permet d'ouvrir l'aide directement dans le navigateur par défaut ;
- `help (mean)` fournit une aide spécifique sur la fonction `mean` ;
- `example (mean)` fournit des exemples d'utilisation de la fonction `mean` ;
- `apropos(mean)` permet d'obtenir la liste de toutes les fonctions qui contiennent le mot "mean" ;
- `q()` permet de quitter `R`.

Si `R` ne reconnaît pas la commande, il retourne un message d'erreur (pas toujours très explicite...). Souvent, il s'agit d'une erreur typographique. Au lieu de retaper l'ensemble de la commande, la flèche de déplacement vers le haut sur le clavier permet de réafficher à l'écran la dernière commande envoyée (et un nouvel appui sur la flèche la commande antérieure, etc.). Il suffit ensuite de corriger l'erreur et d'exécuter à nouveau la commande.

La console ne comporte qu'un éditeur de ligne, ce qui n'est guère pratique lorsque l'on a plusieurs commandes à envoyer. Un petit éditeur de texte est fourni avec `R` accessible à partir du menu `fichier` puis "nouveau script". Les commandes entrées dans cet éditeur s'exécutent avec `CTRL + R` (ou en faisant un clic droit avec la souris). En utilisant `RStudio`, on dispose d'un éditeur bien plus puissant.

3.2 Un langage orienté objet

`R` est un langage orienté objet. Les objets peuvent être (liste non exhaustive) des vecteurs (`vector`), des matrices (`matrix`), des tableaux de données (`data frame`), des listes (`list`) des fonctions (`function`). Ces objets peuvent contenir (liste non exhaustive) des nombres, du texte, des valeurs logiques (`TRUE`, `FALSE`). Chaque objet est désigné par un nom. Ce nom n'est pas limité en nombre de caractères (mais les noms courts sont plus faciles à gérer), ils doivent commencer par une lettre et peuvent comporter les signes typographiques (comme `_` ou `.`), mais pas d'espace[2].

> Le langage `R` est sensible à la casse. Cela signie que les noms `Sexe`, `sexe` et `seXe` vont désigner trois objets différents. Il s'agit donc d'être très vigilant sur ce point !

3.2.1 Les vecteurs

Les vecteurs (`vector`) sont les objets les plus simples de `R`. Ce sont simplement des ensembles de valeurs, d'éléments qui peuvent être de différents types, mais qui sont les plus souvent des valeurs numériques et/ou des éléments alphanumériques.

Par exemple, si dans une étude on a interrogé sept personnes et que l'on a recueilli leur âge, on peut créer le vecteur `age` qui comprend ces sept éléments.

```
age <-  c(68,25,34,28,45,29,54)
```

Cette commande se lit de la manière suivante :

- un nouvel objet est créé dont le nom est `age`[3] ;

[1] Si une commande est entrée sans les parenthèses, `R` retourne des informations génériques sur la commande ou les lignes de code de la commande. C'est sans grande utilité pour l'utilisateur novice, mais avec un peu d'entraînement, on peut arriver à repérer ainsi la logique du langage de programmation.

[2] En fait, on peut même créer des noms d'objet avec des espaces, mais il faut alors systématiquement y faire appel en les inscrivant dans des guillemets (`"`), ce qui complique sérieusement les choses, donc il vaut mieux éviter.

[3] Les lettres accentuées (`â`, `é`, `è`) sont possibles dans les noms d'objets, mais peuvent poser des difficultés de compatibilité dans certains cas très particuliers. Il est donc conseillé de les éviter.

- le signe `<-`[4] signifie que l'on attribue un contenu à l'objet nommé `age` ;
- `c` est une fonction (c'est l'abréviation du verbe "concaténer") qui permet de combiner plusieurs éléments dans un seul objet, un objet de type `vector` ;
- dans la parenthèse, après le nom de la fonction, sont listées (séparées par une virgule)[5], les différentes valeurs qui vont composer le vecteur (ici, les âges des différentes personnes interrogées).

L'objet `age` est alors créé et stocké en mémoire. On peut lister son contenu et appliquer des différentes fonctions. Dans l'exemple suivant, on fait apparaître son contenu à l'écran en tapant simplement son nom[6] et on calcule l'âge moyen avec la fonction `mean`.

```
age
```

```
## [1] 68 25 34 28 45 29 54
```

```
mean (age)
```

```
## [1] 40.42857
```

Il est important de noter dès à présent que les résultats de l'application de fonctions à des objets peuvent eux-mêmes devenir des objets. Même si ce n'est pas très pertinent ici, on peut créer un objet qui va contenir l'âge moyen. Ce nouvel objet sera un vecteur un peu particulier puisqu'il ne contiendra qu'un seul éléments. Les commandes suivantes permettent de :

- créer un nouvel objet appelé age_moy et
- lister son contenu.

```
age_moy <- mean (age)
age_moy
```

```
## [1] 40.42857
```

Outre l'âge de ces sept personnes, imaginons que l'on ait aussi des informations sur leur genre, leur revenu et leur situation matrimoniale[7]. Les commandes suivantes permettent de créer ces trois vecteurs supplémentaires. Après chaque création, on fait afficher le contenu du vecteur pour vérifier que l'on n'a pas fait d'erreur.

```
sexe <-  c("f","m","f","f","m","f","f")
sexe
```

```
## [1] "f" "m" "f" "f" "m" "f" "f"
```

```
revenu <-  c(1200,1100,850,2211,1210,1003,10110)
revenu
```

```
## [1]  1200  1100   850  2211  1210  1003 10110
```

```
matrimo <-  c(4,1,2,2,3,1,1)
matrimo
```

```
## [1] 4 1 2 2 3 1 1
```

On notera le cas particulier du vecteur `sexe`. Tous les éléments non numériques[8] doivent être insérés dans des guillemets.

[4] Simplement le signe inférieur (`<`) suivi, **sans espace**, d'un tiret (`-`).
[5] Comme les virgules servent de séparateur il est indifférent de mettre des espaces avant ou après les virgules : le langage `R` n'en tient pas compte.
[6] C'est en fait un raccourci de la fonction `print` : Les commandes `print (age)` et `age` sont équivalentes.
[7] Codée en 4 catégories : 1 = célibataire ; 2 = marié(e) ou vie maritale ; 3 = divorcé(e) ; 4 = veuf(ve).
[8] Pour être complet, il faudrait écrire "non numériques et non logiques", mais ces subtilités seront abordées plus tardivement.

3.2.2 Les tableaux de données

Dans R, un tableau de données (`data frame`) est une structure rectangulaire de vecteurs, autrement dit, un arrangement de vecteurs ayant chacun le même nombre d'éléments. Concrètement, on peut créer un tableau de données en plaçant chacun des vecteurs précédents dans une colonne. On obtient alors une structure assez classique que l'on trouve dans un tableur par exemple : les différentes lignes correspondent chacun à un individu et les différentes colonnes correspondent chacune à une variable.

La fonction `data.frame` permet de créer un tableau de données. Dans les commandes ci-dessous, on crée d'abord un vecteur supplémentaire (appelé `id`) qui donne un numéro à chaque individu. La fonction `seq` permet de créer une séquence de nombres compris entre les deux valeurs indiquées entre parenthèses et séparées par `:`.

```
id <- seq (1:7)
tab <- data.frame(id, sexe, age, revenu, matrimo)
tab
```

```
##   id sexe age revenu matrimo
## 1  1    f  68   1200       4
## 2  2    m  25   1100       1
## 3  3    f  34    850       2
## 4  4    f  28   2211       2
## 5  5    m  45   1210       3
## 6  6    f  29   1003       1
## 7  7    f  54  10110       1
```

Comme on le voit ci-dessus, ce tableau comporte sept lignes et cinq colonnes. Quelques fonctions de base sont très utiles pour retrouver ces informations lorsque les tableaux sont trop grands pour être affichés à l'écran :

- la fonction `dim` fournit le nombre de lignes et le nombre de colonnes
- la fonction `nrow` fournit seulement le nombre de lignes
- la fonction `ncol` fournit seulement le nombre de colonnes
- la fonction `names` fournit le nom des différentes colonnes (i.e. le nom des variables)

Ces différentes fonctions sont exemplifiées ci-dessous :

```
dim (tab)
```

```
## [1] 7 5
```

```
nrow(tab)
```

```
## [1] 7
```

```
ncol (tab)
```

```
## [1] 5
```

```
names (tab)
```

```
## [1] "id"      "sexe"    "age"     "revenu"  "matrimo"
```

Il existe aussi des fonctions statistiques que l'on peut appliquer directement sur des tableaux de données. L'une des plus versatiles est la fonction `summary` qui permet d'obtenir une description statistique univariée de toutes les colonnes du tableau :

```
summary (tab)
```

```
##        id      sexe       age            revenu         matrimo
##  Min.   :1.0   f:5   Min.   :25.00   Min.   :  850   Min.   :1.0
##  1st Qu.:2.5   m:2   1st Qu.:28.50   1st Qu.: 1052   1st Qu.:1.0
##  Median :4.0         Median :34.00   Median : 1200   Median :2.0
##  Mean   :4.0         Mean   :40.43   Mean   : 2526   Mean   :2.0
##  3rd Qu.:5.5         3rd Qu.:49.50   3rd Qu.: 1710   3rd Qu.:2.5
##  Max.   :7.0         Max.   :68.00   Max.   :10110   Max.   :4.0
```

Comme on l'a déjà vu, les résultats des fonctions peuvent être stockés dans de nouveaux objets. Par exemple, on peut créer un nouveau vecteur comprenant le nom des variables du tableaux de données `tab` :

```
noms_tab <- names (tab)
noms_tab
```

```
## [1] "id"      "sexe"    "age"     "revenu"  "matrimo"
```

3.2.3 Les matrices

Les matrices (`matrix`) sont un cas particulier des tableaux de données. Elles ne peuvent comprendre que des éléments de même type : que des éléments numériques ou que des éléments alphanumériques ou que des éléments logiques. Comme nous aurons recours dans la suite de ce document quasi exclusivement à des tableaux de données, les matrices ne seront pas présentées plus en détail ici.

3.2.4 Les listes

Les listes (`list`) sont des objets très flexibles puisqu'ils peuvent contenir n'importe quels objets, même de nature ou de taille différentes. Par exemple[9], on peut créer un objet appelé `l1` qui va contenir à la fois le tableau de données `tab`, la moyenne de l'âge (`age_moy`) et le nom des variables du tableau de données (`noms_tab`) :

```
l1 <- list (tab, age_moy, noms_tab)
l1
```

```
## [[1]]
##   id sexe age revenu matrimo
## 1  1    f  68   1200       4
## 2  2    m  25   1100       1
## 3  3    f  34    850       2
## 4  4    f  28   2211       2
## 5  5    m  45   1210       3
## 6  6    f  29   1003       1
## 7  7    f  54  10110       1
##
## [[2]]
## [1] 40.42857
##
## [[3]]
## [1] "id"      "sexe"    "age"     "revenu"  "matrimo"
```

[9]Les exemples qui suivent n'ont pas beaucoup d'intérêt et il est peu probable que l'on y ait recours dans des applications réelles. Ils servent juste à illustrer l'extrême flexibilité des objets de type `list`.

Une liste peut même contenir une (ou plusieurs) autres listes. Ainsi l'objet `l2` créé ci-dessous contient à la fois la liste `l1` et le vecteur `age`.

```
l2 <- list (l1, age)
l2
```

```
## [[1]]
## [[1]][[1]]
##   id sexe age revenu matrimo
## 1  1    f  68   1200       4
## 2  2    m  25   1100       1
## 3  3    f  34    850       2
## 4  4    f  28   2211       2
## 5  5    m  45   1210       3
## 6  6    f  29   1003       1
## 7  7    f  54  10110       1
##
## [[1]][[2]]
## [1] 40.42857
##
## [[1]][[3]]
## [1] "id"      "sexe"    "age"     "revenu"  "matrimo"
##
##
## [[2]]
## [1] 68 25 34 28 45 29 54
```

3.3 Les objets dans l'environnement de travail, leur classe, leur structure

A ce stade, de nombreux objets ont déjà été créés et stockés en mémoire. On dit qu'ils sont présents dans l'environnement de travail.

3.3.1 Liste des objets dans l'environnement de travail

La fonction `ls` permet d'avoir la liste des objets qui existent dans l'environnement de travail. Aucun argument n'est nécessaire pour cette fonction, mais les parenthèses sont nécessaires :

```
ls()
```

```
##  [1] "age"      "age_moy"  "id"       "l1"       "l2"       "matrimo"
##  [7] "noms_tab" "revenu"   "sexe"     "tab"      "tabfreq"
```

Les résultats de la fonction `ls` permettent donc de savoir qu'il y a actuellement 11 objets dans l'environnement de travail et de connaître leur nom.

On peut supprimer un objet de l'environnement de travail avec la fonction `rm`[10]. Dans l'exemple suivant, on supprime de l'environnement de travail les vecteurs `age` et `sexe`. Il ne reste alors plus que 9 objets dans l'environnement de travail[11].

```
rm (age, sexe)
ls()
```

[10] Abréviation de *remove*.
[11] Il n'existe aucun moyen d'annuler la suppression d'un objet. La seule possibilité est de le recréer

```
## [1] "age_moy"  "id"       "l1"       "l2"       "matrimo"  "noms_tab"
## [7] "revenu"   "tab"      "tabfreq"
```

Il est très important de noter que si les vecteurs `age` et `sexe` ont bien été supprimés de l'environnement, leur contenu est toujours présent dans le tableau `tab` et dans la liste `l1`. On verra ultérieurement comment accéder à ce contenu, sachant qu'après leur suppression, si on demande d'afficher l'objet supprimé ou si on applique une fonction à un tel objet, on obtient un message d'erreur :

```
sexe
```

```
## Error in eval(expr, envir, enclos): objet 'sexe' introuvable
```

```
mean (age)
```

```
## Error in mean(age): objet 'age' introuvable
```

Ces messages d'erreur sont tout à fait logiques, puisque les objets n'existent plus dans l'environnement de travail.

3.3.2 Les classes des objets

On l'a vu, il existe différentes classes d'objets. Quand l'environnement comporte de nombreux objets ou que certains objets ont été créés automatiquement par certaines analyses statistiques, on ne sait plus forcément à quel type renvoie tel objet. La fonction `class` permet de connaître le type d'objet :

```
class (revenu)
```

```
## [1] "numeric"
```

```
class (tab)
```

```
## [1] "data.frame"
```

```
class (l1)
```

```
## [1] "list"
```

3.3.3 La structure des objets

Des objets ont un contenu très simple (un vecteur avec des éléments numériques par exemple) ; d'autres peuvent avoir un contenu beaucoup plus complexe (une liste peut contenir des vecteurs, des matrices, des tableaux de données... et d'autres listes). La fonction `str` permet d'obtenir un résumé de la structure d'un objet.

Voyons, par exemple, la structure de l'objet `tab` :

```
str (tab)
```

```
## 'data.frame':    7 obs. of  5 variables:
##  $ id     : int  1 2 3 4 5 6 7
##  $ sexe   : Factor w/ 2 levels "f","m": 1 2 1 1 2 1 1
##  $ age    : num  68 25 34 28 45 29 54
##  $ revenu : num  1200 1100 850 2211 1210 ...
##  $ matrimo: num  4 1 2 2 3 1 1
```

Les résultats de cette commande nous indique donc que:

- l'objet `tab` est de type `data frame` ;
- il comporte 7 lignes ("obs.") et 5 colonnes ("variables") ;
- Les différentes colonnes prennent pour noms : "id", "sexe", "age", "revenu", "matrimo" :
  - La colonne `id` est elle-même un vecteur ne comprenant que des valeurs entières ("int" comme "*integer*") ;
  - La colonne `revenu` est un vecteur contenant des valeurs numériques ("num") ;
  - La colonne `sexe` est un vecteur de type `Factor`, c'est-à-dire un vecteur qui sera considéré comme une variable qualitative[12] ;
  - etc.

La structure de l'objet `l1` est la suivante :

```
str (l1)
```

```
## List of 3
##  $ :'data.frame':    7 obs. of  5 variables:
##   ..$ id     : int [1:7] 1 2 3 4 5 6 7
##   ..$ sexe   : Factor w/ 2 levels "f","m": 1 2 1 1 2 1 1
##   ..$ age    : num [1:7] 68 25 34 28 45 29 54
##   ..$ revenu : num [1:7] 1200 1100 850 2211 1210 ...
##   ..$ matrimo: num [1:7] 4 1 2 2 3 1 1
##  $ : num 40.4
##  $ : chr [1:5] "id" "sexe" "age" "revenu" ...
```

Il s'agit d'une liste comprenant trois éléments :

- un tableau de données (avec le détail de la structure du tableau) ;
- un vecteur numérique ne comprenant qu'un seul élément ;
- un autre vecteur, alphanumérique celui-ci, comprenant 5 éléments.

3.3.4 L'environnement de travail avec `RStudio`

La visualisation de l'environnement de travail avec le logiciel `RStudio` (Figure 3.2) est grandement facilitée grâce à la présence d'une fenêtre (en haut à droite, sous l'onglet "environnement") qui permet d'afficher le contenu de l'environnement de travail.

[12] Comme la colonne `sexe` contient des caractères alphanumériques, le logiciel la reconnait automatiquement comme une variable qualitative. La colonne `matrimo` ne contient quant à elle que des valeurs numériques. Elle est pourtant aussi, conceptuellement, une variable qualitative, mais le logiciel ne peut la reconnaître automatiquement comme telle. La fonction `factor` permet d'indiquer explicitement qu'il s'agit d'une variable qualitative. On verra des exemples de son utilisation ultérieurement.

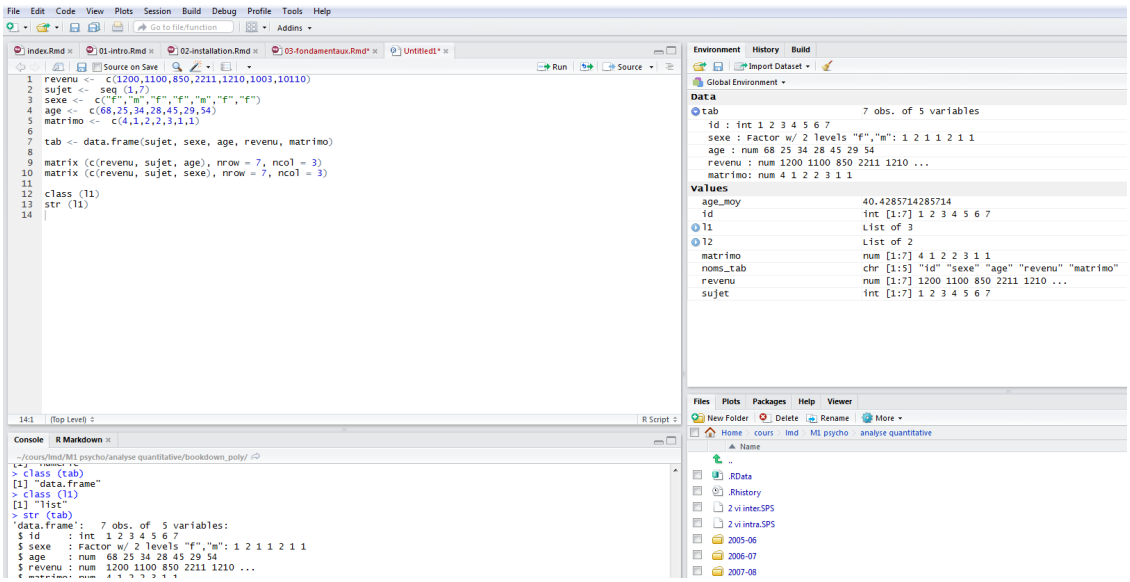


Figure 3.2: Visualisation de l'environnement de travail avec RStudio

La petite flèche qui se trouve à côté de certains objets permet d'en visualiser la structure. Dans l'exemple de la figure 3.2, la structure de l'objet `tab` a été déroulée, mais pas celle des objets `l1` et `l2`.

3.3.5 Sauvegarder l'environnement de travail

L'environnement de travail se compose donc de plusieurs objets créés lors d'une session. Cet environnement peut être sauvegardé pour être utilisable ultérieurement.

Cette procédure sauvegarde bien tout l'environnement de travail et dans cet environnement de travail, il peut y avoir plusieurs tableaux de données. On peut aussi exporter un tableau de données (un à la fois) pour le lire dans un tableur par exemple. C'est la fonction `write` qu'il faut alors utiliser. On consultera l'aide sur cette fonction pour en savoir plus : `help (write)`[13].

3.3.5.1 Avec le logiciel `R`

Avec le logiciel `R`, l'environnement de travail se sauvegarde par l'intermédiaire du menu `Fichier`[14] puis : "Sauver l'environnement de travail". La fenêtre qui s'ouvre permet de donner un nom au fichier contenant l'environnement. Ce fichier est suffixé par `.RData` (par exemple : `essai.RData`). Pour retrouver les données lors d'une session ultérieure, il suffit de sélectionner "charger l'environnement de travail" dans le menu `fichier`, puis se placer dans le répertoire approprié et choisir le fichier souhaité (dans notre exemple, le fichier `essai.RData`).

3.3.5.2 Avec le logiciel `RStudio`

Avec le logiciel `RStudio`, dans la fenêtre en haut à droite, onglet "environnement", on trouve deux icones (entourées en rouge dans la figure 4.1). La seconde icone permet de sauvegarder l'environnement de travail existant ; la première permet d'ouvrir un environnement de travail sauvegardé au préalable.

[13] L'exportation avec la fonction `write` se décline de différentes manières. Pour l'exportation d'un fichier au format `csv` facilement lisible par tout tableur, on pourra préférer la fonction `write.csv`.

[14] Cela correspond à la fonction `save.image`.

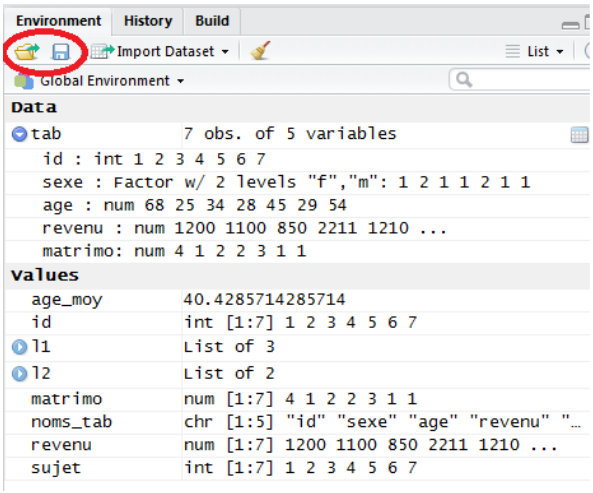


Figure 3.3: Les deux icones de RStudio pour sauvegarder et charger l'environnement de travail

Chapter 4

Importer un fichier de données dans R

4.1 Les formats de fichier

Dans le chapitre précédent, on a vu comment construire un tableau de données directement dans R. Cette manière de procéder n'est pas la plus efficace. Il est beaucoup plus aisé de construire son fichier de données dans un tableur, qui offre une plus grande ergonomie et de l'importer ensuite dans R. Le logiciel R reconnaît plusieurs formats de fichiers lors de l'importation[1], mais le plus simple est de sauvegarder son fichier au format csv[2].

4.2 La référence à un répertoire de travail

Pour importer un fichier au format csv dans R, la fonction est : read.csv. Il faut indiquer entre guillemets le nom du fichier à importer avec son chemin complet (répertoires et sous-répertoires) ce qui donne des commandes longues et complexes[3], sources de potentielles fautes de frappe[4]. Il est donc préférable de spécifier au préalable le répertoire dans lequel se trouve le fichier. Ce répertoire deviendra le répertoire de travail et R s'y positionnera automatiquement et par défaut lors de toute référence à un fichier (que ce soit lors de sauvegarde ou d'ouverture).

La spécification d'un répertoire de travail se fait avec la fonction setwd[5], suivie entre guillemets du chemin d'accès. Sous windows, cela peut prendre la forme, par exemple de : `setwd("~/cours/lmd/M1 psycho/analyse quantitative/poly")`. Avec RStudio, la spécification du répertoire de travail est facilitée en utilisant le menu session puis "*set working directory*" puis "*choose directory*".

[1] Il existe plusieurs paquets permettant de faciliter l'importation de fichiers de données. Par exemple, les paquets gdata (Warnes et al. (2017)), XLConnect (Mirai Solutions GmbH (2017)), xlsx (Dragulescu (2014)) pour les fichiers Excel ; le paquet foreign (R Core Team (2017a)) pour les fichiers SPSS, SAS, Systat, Minitab, dBase, ...

[2] Tous les logiciels de type tableur (ex : Excel, OpenOffice...) ou base de données (ex : Access) offrent la possibilité de sauvegarder un fichier au format csv. En général, dans le menu "fichier", il faut sélectionner "enregistrer sous" puis modifier, dans une liste déroulante, le type de fichier (choisir csv). Certains logiciels requièrent le choix du caractère qui va séparer les différentes colonnes. Il est alors conseillé de prendre le caractère ;.

[3] Comme, par exemple : `read.csv ("C:/Users/Jean Luc Kop/Documents/cours/lmd/M1 psycho/analyse quantitative/notes250.csv")`.

[4] Une alternative consiste à utiliser la fonction file.choose qui permet d'ouvrir l'explorateur de fichiers. Il faut alors combiner les deux fonctions dans une même commande : `read.csv (file.choose())`.

[5] Il faut comprendre "*set working directory*".

4.3 Un exemple d'importation de fichier

Le fichier `notes250.csv` (téléchargement) est un fichier `csv` contenant les résultats fictifs de 250 étudiants inscrits dans un diplôme imaginaire qui fonctionnerait selon les règles suivantes :

- il y a deux unités d'enseignement (UE) comprenant chacune trois cours (appelés respectivement cours1, cours2, cours3 et cours4, cours5, cours6) ;
- dans la première UE, les trois cours sont obligatoires ;
- dans la seconde UE, le cours4 est obligatoire alors que les étudiants doivent choisir entre le cours5 et le cours6 ;
- pour obtenir son diplôme, l'étudiant doit avoir à la fois une moyenne supérieure à 10 dans l'UE1 et dans l'UE2 (les notes à l'intérieur des blocs sont équipondérées) ;
- une mention est attribuée, le cas échéant, sur la base d'une moyenne des notes aux deux UE en attribuant le coefficient 1,5 à la première UE et le coefficient 1 à la seconde UE.

Les variables du fichier sont les suivantes :

- "num" : numéro de l'étudiant (de 1 à 250)
- "sexe" : sexe de l'étudiant (1 = masculin ; 2 = féminin)
- de "cours1" à "cours6" (notes aux 6 cours ; -8 = absent à l'examen ; pour cours5 et cours6, -9 = non concerné par l'examen)

L'importation de ce fichier se fera donc à partir de la commande suivante[6] :

```
n250 <- read.csv2 ("notes250.csv")
```

La fonction `read.csv2` est un raccourci de la commande générale `read.csv`. Elle comporte plusieurs options par défaut qui sont adaptées au fichier que l'on importe et notamment :

- le fait que sur la première ligne du fichier de données, on trouve le nom des variables (option `header = TRUE`) ;
- le séparateur entre les colonnes est un `;` (option `sep = ";"`) ;
- le caractère décimal est une virgule (option `sep = ","`).

Il est très important de vérifier que l'importation est correcte : les fonctions `dim`, `head` et `tail` permettent d'obtenir respectivement le nombre de lignes et de colonnes du fichier, l'affiche des premières lignes et l'affichage des dernières lignes, comme dans les exemples suivants /

```
dim (n250)
```

```
## [1] 250   8
```

```
head (n250)
```

```
##   num sexe cours1 cours2 cours3 cours4 cours5 cours6
## 1   1    2     11     13     11     15     13     -9
## 2   2    1     -8     14     13     13     -9     -8
## 3   3    2      9     12      9      9     11     -9
## 4   4    2      9     12     11     13     -9     11
## 5   5    1     10     10     -8     10      8     -9
## 6   6    2     10     12      9      8     -9      9
```

```
tail (n250)
```

```
##     num sexe cours1 cours2 cours3 cours4 cours5 cours6
```

[6] On suppose ici que le fichier de données est stocké dans le répertoire de travail.

```
## 245 245     2      -8      -8       4       9       6      -9
## 246 246     1      11      11      10      10      -9      11
## 247 247     1       8       7      15       9      14      -9
## 248 248     1       9       9      13      12      -9      15
## 249 249     2      -8      13       8       9       7      -9
## 250 250     2       9      13      13      11      -9       7
```

On notera donc que l'on a attribué au fichier de données importé le nom de `n250`. Celui-ci est maintenant le nom d'un objet dans l'environnement de travail de `R` : c'est un objet de type `data frame`. Avec `RStudio`, cet objet apparaît dans la fenêtre de l'environnement de travail (en haut à droite), avec des indications sur son format. Un clic sur l'icone qui se trouve à droite de l'objet permet de visualiser son contenu de la même manière que dans un tableur (Figure 4.1).

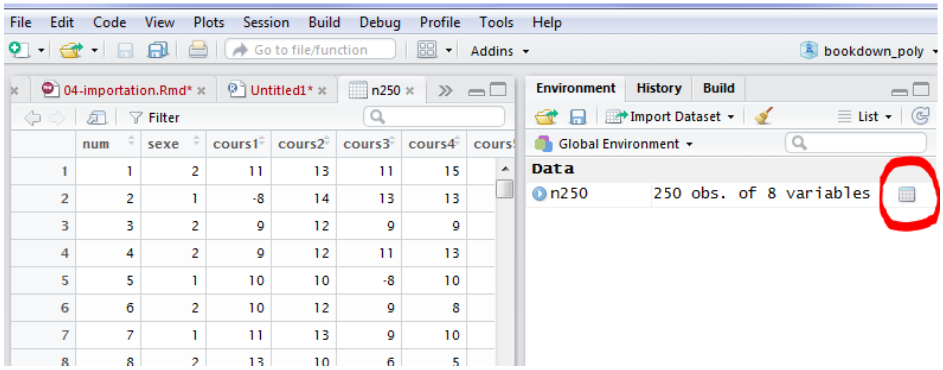


Figure 4.1: Visualisation du fichier importé dans RStudio

4.4 Des premières statistiques sur le fichier de données

La fonction `summary` permet d'obtenir des statistiques de base[7] de l'ensemble des variables d'un tableau de données. Outre les statistiques proprement dites, c'est aussi une autre manière de vérifier que l'importation a été réalisée correctement.

```
summary (n250)
```

```
##       num              sexe           cours1           cours2
##  Min.   :  1.00   Min.   :1.000   Min.   :-8.000   Min.   :-8.00
##  1st Qu.: 63.25   1st Qu.:2.000   1st Qu.: 8.000   1st Qu.:10.00
##  Median :125.50   Median :2.000   Median :10.000   Median :12.00
##  Mean   :125.50   Mean   :1.812   Mean   : 9.276   Mean   :10.98
##  3rd Qu.:187.75   3rd Qu.:2.000   3rd Qu.:11.000   3rd Qu.:14.00
##  Max.   :250.00   Max.   :2.000   Max.   :16.000   Max.   :20.00
##      cours3          cours4          cours5          cours6
##  Min.   :-8.00   Min.   :-8.0   Min.   :-9.00   Min.   :-9.000
##  1st Qu.: 8.00   1st Qu.: 6.0   1st Qu.:-9.00   1st Qu.:-9.000
##  Median :11.00   Median : 9.0   Median :-8.00   Median :-8.000
##  Mean   :10.01   Mean   : 8.2   Mean   : 0.08   Mean   : 0.788
##  3rd Qu.:14.00   3rd Qu.:11.0   3rd Qu.:10.00   3rd Qu.:11.000
##  Max.   :20.00   Max.   :18.0   Max.   :14.00   Max.   :19.000
```

[7] Valeur minimale, 1er quartile (25e décile), médiane (50e décile), moyenne, 3e quartile (75e centile) et valeur maximale.

Deux points importants sont à noter dans ces premiers résultats :

- le logiciel calcule des statistiques qui n'ont aucun sens pour certaines variables (la moyenne ou la médiane pour le sexe ou pour le numéro de l'étudiant par exemple) ;
- comme l'illustrent les moyennes absurdes calculées pour les deux derniers cours, R assimile les codes -8 et -9 utilisés pour repérer les étudiants qui n'ont pas passé les examens à des valeurs numériques comme les autres.

Le premier point peut être résolu en indiquant que les variables "num"" et "sexe" sont des variables nominales (des facteurs (factor) dans la terminologie de R)[8] :

```
n250$num <-  factor (n250$num)
n250$sexe <-  factor (n250$sexe)
summary (n250)
```

```
##       num      sexe        cours1           cours2          cours3
##  1      :  1   1: 47   Min.   :-8.000   Min.   :-8.00   Min.   :-8.00
##  2      :  1   2:203   1st Qu.: 8.000   1st Qu.:10.00   1st Qu.: 8.00
##  3      :  1           Median :10.000   Median :12.00   Median :11.00
##  4      :  1           Mean   : 9.276   Mean   :10.98   Mean   :10.01
##  5      :  1           3rd Qu.:11.000   3rd Qu.:14.00   3rd Qu.:14.00
##  6      :  1           Max.   :16.000   Max.   :20.00   Max.   :20.00
##  (Other):244
##      cours4          cours5          cours6
##  Min.   :-8.0   Min.   :-9.00   Min.   :-9.000
##  1st Qu.: 6.0   1st Qu.:-9.00   1st Qu.:-9.000
##  Median : 9.0   Median :-8.00   Median :-8.000
##  Mean   : 8.2   Mean   : 0.08   Mean   : 0.788
##  3rd Qu.:11.0   3rd Qu.:10.00   3rd Qu.:11.000
##  Max.   :18.0   Max.   :14.00   Max.   :19.000
##
```

On verra plus tard comment gérer le second point. Avant cela, il est nécessaire de comprendre comment R permet d'accéder à des parties spécifiques d'un tableau de données ou d'un vecteur : c'est la logique de l'indexation.

[8] L'utilisation du signe $ dans n250$num signifie que la variable "num" se trouve à l'intérieur du tableau de données "n250". La logique de cette notation est présentée plus en détail dans le chapitre suivant.

Chapter 5

L'indexation des objets

5.1 Le principe de l'indexation

A ce stade, l'environnement de travail comporte neuf objets (Figure 5.1) :

- deux tableaux de données (`n250` et `tab`) ;
- deux listes (`l1` et `l2`) ;
- cinq vecteurs

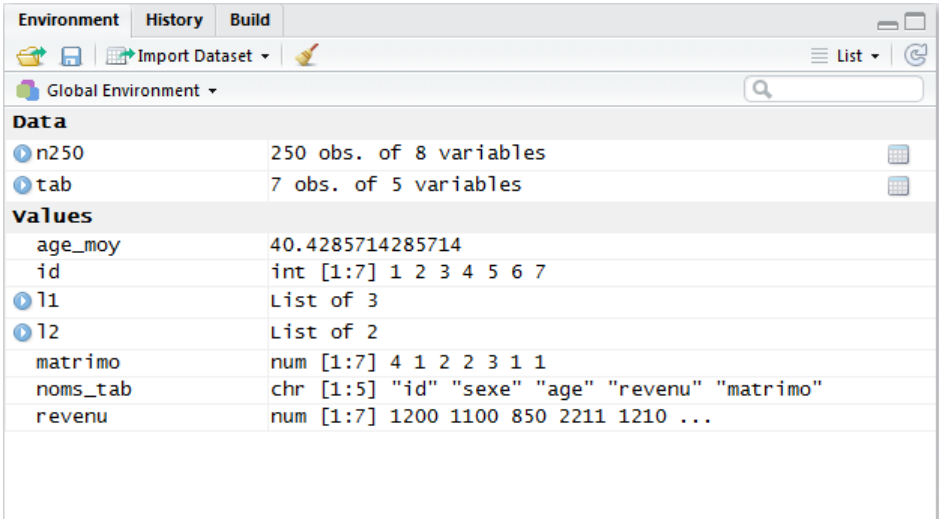


Figure 5.1: Les neuf objets de l'environnement de travail

En écrivant le nom d'un objet, on accède à la totalité de son contenu. Voici par exemple le contenu du vecteur `revenu`.

```
revenu
```

```
## [1]  1200  1100   850  2211  1210  1003 10110
```

Il est souvent nécessaire d'accèder à une partie seulement de l'objet (lorsqu'on veut, par exemple, faire des analyses seulement sur les femmes, ou calculer la moyenne des étudiants qui ont pris en option tel cours). C'est le principe de l'**indexation**.

5.1.1 L'indexation d'un vecteur

L'indexation d'un vecteur se réalise à partir des signes `[]`. Par exemple, la commande suivante permet d'extraire du vecteur `revenu` le quatrième élément :

```
revenu [4]
```

```
## [1] 2211
```

On peut bien sûr extraire plusieurs éléments :

```
revenu [1:3]  # trois premiers éléments
```

```
## [1] 1200 1100  850
```

```
revenu [c(1,3,5)]  # éléments 1, 3 et 5
```

```
## [1] 1200  850 1210
```

Le signe `-` signifie que l'on veut retenir tous les éléments sauf ceux indiqués :

```
revenu [-c(1,3,5)]
```

```
## [1]  1100  2211  1003 10110
```

Plutôt que sur des positions dans le vecteur (éléments 1, 3 et 5 comme dans le dernier exemple ci-dessus), l'indexation peut porter sur des conditions logiques. Par exemple, si l'on veut sélectionner dans le vecteur `revenu` uniquement les revenus supérieurs à 1200, on écrira :

```
revenu [revenu > 1200]
```

```
## [1]  2211  1210 10110
```

Cette commande se lit de la manière suivante : *dans le vecteur revenu, retiens tous les éléments ayant une valeur supérieure à 1200*. Il est très important de noter qu'il faut indiquer explicitement à l'intérieur des crochets le nom de l'objet sur lequel porte la condition logique (et donc écrire deux fois "revenu" dans la commande). Les raisons de cette syntaxe apparaitront plus clairement ultérieurement.

Les principaux symboles utilisables sont les suivants :

- `==` égalité (**notez bien le fait qu'il faut répéter deux fois le signe égal**)
- `!=` différent
- `<` strictement inférieur
- `>` strictement supérieur
- `<=` inférieur ou égal
- `>=` supérieur ou égal
- `is.na(x)` valeur manquante
- `&` ET logique
- `|` OU logique
- `!` NON logique

5.1.2 L'indexation d'un tableau de données

L'indexation d'un tableau de données suit la même logique, sauf qu'il faut indiquer maintenant deux indices : les lignes (en premier) et les colonnes (en second). Ainsi, pour obtenir l'élément qui se trouve à la quatrième ligne et à la troisième colonne du tableau `tab`, on écrira[1] :

[1] Il s'agit, en l'occurrence, de l'âge du sujet numéro 4.

```
tab[4,3]
```

```
## [1] 28
```

L'indexation d'un tableau de fait entre crochets ; on indique d'abord la ou les lignes à extraire, puis la ou les colonnes souhaitées selon la structure générale suivante : `tableau[ligne(s),colonnes(s)]`

Pour extraire une ligne complète, le plus simple est de laisser vide l'indice correspondant aux lignes. C'est la même chose pour les colonnes. Par exemple :

```
tab [2,]  # toutes les variables pour le sujet numéro 2
```

```
##   id sexe age revenu matrimo
## 2  2    m  25   1100       1
```

```
tab [,3]  # les âges de tous les sujets
```

```
## [1] 68 25 34 28 45 29 54
```

Notez que plutôt que le numéro de la variable, on peut indiquer son nom entre guillemets. La commande suivante donnera donc le même résultat que la commande précédente :

```
tab [,"age"]
```

```
## [1] 68 25 34 28 45 29 54
```

On peut faire des extractions de plusieurs lignes ou de plusieurs colonnes :

```
tab [,c(1:3,5)]  # toutes les lignes mais uniquement les colonnes 1, 3 et 5
```

```
##   id sexe age matrimo
## 1  1    f  68       4
## 2  2    m  25       1
## 3  3    f  34       2
## 4  4    f  28       2
## 5  5    m  45       3
## 6  6    f  29       1
## 7  7    f  54       1
```

```
tab [c(2,4,6),c(1,5)]  # les lignes 2, 4 et 6 avec uniquement les colonnes 1 et 5
```

```
##   id matrimo
## 2  2       1
## 4  4       2
## 6  6       1
```

On peut, ici aussi, remplacer les numéros des colonnes occupées par les variables avec leur nom :

```
tab [,c("age", "sexe", "revenu")]
```

```
##   age sexe revenu
## 1  68    f   1200
## 2  25    m   1100
## 3  34    f    850
## 4  28    f   2211
## 5  45    m   1210
## 6  29    f   1003
```

```
## 7  54    f  10110
```

Si on se réfère à la liste des objets dans l'environnement de travail, il est très important (Figure 5.1) de comprendre que la variable "âge" qui figure dans le tableau `tab` n'existe pas indépendamment du tableau dans lequel elle se trouve. Ainsi, il n'est pas possible d'afficher directement le contenu de la variable "age" en écrivant simplement son nom, car cet objet n'existe pas en tant que tel dans l'environnement de travail :

```
age
```

```
## Error in eval(expr, envir, enclos): objet 'age' introuvable
```

C'est pourquoi, on vient de le voir, le contenu de la variable "âge" est accessible en l'extrayant du tableau avec la logique de l'indexation :

```
tab  [,"age"]
```

```
## [1] 68 25 34 28 45 29 54
```

Comme l'écriture de commandes avec les crochets est parfois un peu lourde, il est possible d'utiliser un raccourci, grâce au symbole `$`. Ainsi il est aussi possible d'extraire la variable "age" en écrivant :

```
tab$age
```

```
## [1] 68 25 34 28 45 29 54
```

Grâce à cette astuce, on peut écrire plus simplement des extractions complexes. En voici quelques exemples :

```
# extraire toutes les variables pour les sujets dont l'âge est supérieur à 40 ans
tab [(tab$age > 40),]
```

```
##   id sexe age revenu matrimo
## 1  1    f  68   1200       4
## 5  5    m  45   1210       3
## 7  7    f  54  10110       1
```

```
# extraire toutes les variables pour les sujets dont l'âge est supérieur à 40 ans
# et qui sont de sexe féminin
tab [(tab$age > 40 & tab$sexe == "f"),]
```

```
##   id sexe age revenu matrimo
## 1  1    f  68   1200       4
## 7  7    f  54  10110       1
```

```
# extraire toutes les variables pour les sujets dont l'âge est supérieur à 40 ans
# OU qui sont de sexe féminin
tab [(tab$age > 40 | tab$sexe == "f"),]
```

```
##   id sexe age revenu matrimo
## 1  1    f  68   1200       4
## 3  3    f  34    850       2
## 4  4    f  28   2211       2
## 5  5    m  45   1210       3
## 6  6    f  29   1003       1
## 7  7    f  54  10110       1
```

Il est à nouveau très important de noter que l'on ne peut pas se contenter d'écrire : `tab[age > 40,]`. En procédant ainsi, le logiciel cherche un objet "age" dans l'environnement de travail

général et cet objet n'existe pas (Figure 5.1). La variable "age" n'existe qu'à l'intérieur du tableau de données. Il faut donc l'indiquer explicitement.

```
tab[age > 40,]
```

```
## Error in `[.data.frame`(tab, age > 40, ): objet 'age' introuvable
```

Il peut toutefois arriver, comme c'est le cas ici, que certains objets présents dans l'environnement de travail prennent le même nom que des variables dans un tableau de données. C'est pourquoi, dans la configuration de l'environnement de travail que nous avons construit jusqu'à présent, les deux commandes suivantes donneront le même résultat.

```
tab [revenu > 1100, ]
```

```
##   id sexe age revenu matrimo
## 1  1    f  68   1200       4
## 4  4    f  28   2211       2
## 5  5    m  45   1210       3
## 7  7    f  54  10110       1
```

```
tab [tab$revenu > 1100, ]
```

```
##   id sexe age revenu matrimo
## 1  1    f  68   1200       4
## 4  4    f  28   2211       2
## 5  5    m  45   1210       3
## 7  7    f  54  10110       1
```

Mais c'est une exception et de manière générale il est très dangereux d'avoir dans son environnement de travail des objets qui portent le même nom que des variables dans un tableau. Avoir le même nom ne signifie pas que le contenu est le même. Et si le contenu de l'objet `revenu` n'est pas strictement identique au contenu de la variable "revenu" dans le tableau de données, on s'expose à de graves déconvenues. Il est donc plus prudent de veiller à ne pas utiliser les mêmes noms pour désigner les objets que les noms utilisés pour désigner les variables à l'intérieur d'un tableau de données.

Signalons enfin, pour terminer ce paragraphe sur l'indexaction des tableaux de données, que toute extraction peut devenir elle-même un nouvel objet. Il suffit de lui donner un nom. Ainsi, si on veut créer un tableau de données ne comprenant que les personnes de sexe féminin (appelé `tabf` dans l'exemple ci-dessous), on pourra écrire :

```
tabf <- tab[tab$sexe == "f", ]
tabf
```

```
##   id sexe age revenu matrimo
## 1  1    f  68   1200       4
## 3  3    f  34    850       2
## 4  4    f  28   2211       2
## 6  6    f  29   1003       1
## 7  7    f  54  10110       1
```

On peut ainsi facilement créer de nouveaux objets à partir de l'extraction de contenu d'objets existants. Cette possibilité n'est pas réservée aux tableaux de données : elle est tout aussi simple pour des vecteurs, des matrices ou des listes.

5.1.3 L'indexation d'une liste

Rappelons qu'une liste est un objet comprenant des éléments de différents types. Dans l'environnement de travail actuelle, la liste `l1` comprend un tableau de données et deux vecteurs ; la liste `l2` comprend une liste (composée elle-même d'un tableau de données et de deux vecteurs) et un vecteur.

```
str (l1)
```

```
## List of 3
##  $ tableau :'data.frame':    7 obs. of  5 variables:
##   ..$ id     : int [1:7] 1 2 3 4 5 6 7
##   ..$ sexe   : Factor w/ 2 levels "f","m": 1 2 1 1 2 1 1
##   ..$ age    : num [1:7] 68 25 34 28 45 29 54
##   ..$ revenu : num [1:7] 1200 1100 850 2211 1210 ...
##   ..$ matrimo: num [1:7] 4 1 2 2 3 1 1
##  $ m.age   : num 40.4
##  $ noms.var: chr [1:5] "id" "sexe" "age" "revenu" ...
```

```
str (l2)
```

```
## List of 2
##  $ :List of 3
##   ..$ :'data.frame': 7 obs. of  5 variables:
##   .. ..$ id     : int [1:7] 1 2 3 4 5 6 7
##   .. ..$ sexe   : Factor w/ 2 levels "f","m": 1 2 1 1 2 1 1
##   .. ..$ age    : num [1:7] 68 25 34 28 45 29 54
##   .. ..$ revenu : num [1:7] 1200 1100 850 2211 1210 ...
##   .. ..$ matrimo: num [1:7] 4 1 2 2 3 1 1
##   ..$ : num 40.4
##   ..$ : chr [1:5] "id" "sexe" "age" "revenu" ...
##  $ : num [1:7] 68 25 34 28 45 29 54
```

Le principe général d'indexation d'une liste consiste à utiliser les crochets doubles `[[ ]]` et à indiquer le ou les numéros des éléments à extraire à l'intérieur de ces crochets. Ainsi, les commandes suivantes permettront d'extraire d'abord le premier élément de la liste `l1` (le tableau de données) puis le second élément de la liste `l2`.

```
l1 [[1]]  # extraction du premier élément de la liste l1
```

```
##   id sexe age revenu matrimo
## 1  1    f  68   1200       4
## 2  2    m  25   1100       1
## 3  3    f  34    850       2
## 4  4    f  28   2211       2
## 5  5    m  45   1210       3
## 6  6    f  29   1003       1
## 7  7    f  54  10110       1
```

```
l2 [[2]]  # extraction du deuxième élément de la liste l2
```

```
## [1] 68 25 34 28 45 29 54
```

Les crochets et les doubles crochets peuvent se combiner. Ainsi, pour extraire la troisième ligne de la quatrième variable du tableau de données qui est le premier élément de la liste `l1`, il faut écrire :

```
l1 [[1]] [3,4]
```

```
## [1] 850
```

Avec la liste `l2` qui contient elle-même une liste, le principe est le même. Pour extraire la même information que ci-dessus, on écrira donc :

```
l2 [[1]] [[1]] [3,4]
```

```
## [1] 850
```

Après le nom de la liste, le premier `[[1]]` permet d'extraire le premier élément de la liste qui est une liste. Cette liste comprend notamment un tableau de données (le premier élément) qui est extrait avec le second `[[1]]`. Enfin, le recours à des crochets simples (`[3,4]`) suffit à extraire la troisième ligne et la quatrième colonne du tableau de données.

Cette écriture peut paraître assez complexe au premier abord. Elle est toutefois extrêmement puissante et se laisse décoder facilement pour peu que l'on connaisse la structure de la liste, ce qui est facile avec la fonction `str`.

Les listes sont souvent des objets qui stockent les résultats d'analyses statistiques dans `R`. Elles stockent alors des informations de nature différente. Par exemple, la fonction `cor` (corrélation) produit une liste dans laquelle on trouve notamment la valeur de la corrélation, la significativité, l'intervalle de confiance, etc. Généralement, les différents éléments de la liste ont un nom et il n'est pas nécessaire de les extraire avec l'indexation numérique que nous venons de voir. L'utilisation du symbole `$` est suffisant, comme avec les tableaux de données.

Reprenons la liste `l1` et attribuons des noms à ses trois éléments :

```
names (l1) <- c("tableau", "m.age", "noms.var")
str (l1)
```

```
## List of 3
##  $ tableau :'data.frame':    7 obs. of  5 variables:
##   ..$ id     : int [1:7] 1 2 3 4 5 6 7
##   ..$ sexe   : Factor w/ 2 levels "f","m": 1 2 1 1 2 1 1
##   ..$ age    : num [1:7] 68 25 34 28 45 29 54
##   ..$ revenu : num [1:7] 1200 1100 850 2211 1210 ...
##   ..$ matrimo: num [1:7] 4 1 2 2 3 1 1
##  $ m.age   : num 40.4
##  $ noms.var: chr [1:5] "id" "sexe" "age" "revenu" ...
```

Lorsque les éléments de la liste sont désignés par des noms (qui apparaissent après le symbole `$` dans le résultat de la fonction `str`), les deux commandes suivantes sont équivalentes :

```
l1 [[2]]  # extraction par le numéro de l'élément
```

```
## [1] 40.42857
```

```
l1$m.age  # extraction par le nom de l'élément
```

```
## [1] 40.42857
```

Il est bien sûr possible de combiner l'indexation avec les crochets et l'indexation avec les noms d'objets, comme le montre l'exemple suivant :

```
l1$tableau [3,4]
```

```
## [1] 850
```

5.2 Le cas particulier des données manquantes

5.2.1 La déclaration des données manquantes

Traditionnellement, lors de la saisie de données, on réserve un ou plusieurs codes spécifiques pour les données manquantes (dues aux non-réponses, aux refus de réponse, aux questions qui ne s'appliquent pas...), par exemple les codes 9, -1, -9 ou tout autre code non ambigu en fonction de la nature de la variable.

Dans certaines analyses, il est nécessaire que `R` reconnaisse qu'il s'agit de données manquantes, c'est-à-dire d'éléments qui ne doivent pas être prises en compte dans les calculs : c'est le symbole `NA`[2] qui repère les données manquantes.

Considérons l'exemple suivant (il s'agit des 10 premiers sujets du tableau `n250`) comprenant des notes à des examens d'étudiants. Le code `-8` désigne des étudiants non inscrits au cours et le code `-9` désigne des étudiants inscrits mais qui n'ont pas passé l'examen.

```
n250 [1:10,]
```

```
##    num sexe cours1 cours2 cours3 cours4 cours5 cours6
## 1    1    2     11     13     11     15     13     -9
## 2    2    1     -8     14     13     13     -9     -8
## 3    3    2      9     12      9      9     11     -9
## 4    4    2      9     12     11     13     -9     11
## 5    5    1     10     10     -8     10      8     -9
## 6    6    2     10     12      9      8     -9      9
## 7    7    1     11     13      9     10     14     -9
## 8    8    2     13     10      6      5     -9      7
## 9    9    2      7     13      4      4     11     -9
## 10  10    2     10     11     12      2     -9      6
```

On a déjà vu que tout calcul statistique sur ces données conduit évidemment à des résultats absurdes. Par exemple :

```
mean (n250$cours6)
```

```
## [1] 0.788
```

Il est donc absolument nécessaire de déclarer les codes utilisés pour repérer des informations manquantes en tant que données manquantes (*NA*). Les commandes suivantes permettent de déclarer, pour chacune des variables "cours" les codes de données manquantes qui ont été utilisés[3] :

```
n250$cours1 [n250$cours1 ==  -8] <-  NA
n250$cours2 [n250$cours2 ==  -8] <-  NA
n250$cours3 [n250$cours3 ==  -8] <-  NA
n250$cours4 [n250$cours4 ==  -8] <-  NA
n250$cours5 [n250$cours5 ==  -8 | n250$cours5 ==  -9] <-  NA
n250$cours6 [n250$cours6 ==  -8 | n250$cours6 ==  -9] <-  NA
```

Lorsqu'on affiche les 10 premières lignes du tableau de données, on remarque que les valeurs "-8" et "-9" ont été remplacées par "NA" :

```
n250 [1:10,]
```

```
##    num sexe cours1 cours2 cours3 cours4 cours5 cours6
## 1    1    2     11     13     11     15     13     NA
```

[2] Comme *Not Applicable*. On trouve aussi parfois dans certaines analyses le symbole `NaN` (*Not a Number*) qui est produit lors d'un calcul impossible, comme la division d'un nombre par zéro par exemple.

[3] On pourrait condenser ces 6 lignes de commandes en une seule, grâce à une écriture plus compacte : `n250 [,3:8] [n250[,3:8] == -8 | n250[,3:8] == -9] <- NA`.

```
## 2    2    1     NA     14     13     13     NA     NA
## 3    3    2      9     12      9      9     11     NA
## 4    4    2      9     12     11     13     NA     11
## 5    5    1     10     10     NA     10      8     NA
## 6    6    2     10     12      9      8     NA      9
## 7    7    1     11     13      9     10     14     NA
## 8    8    2     13     10      6      5     NA      7
## 9    9    2      7     13      4      4     11     NA
## 10  10    2     10     11     12      2     NA      6
```

5.2.2 Les calculs statistiques en présence de données manquantes

Les calculs statistiques fournissent maintenant des résultats qui font sens. Par exemple, la fonction `summary` ci-dessous, qui calcule des statistiques en excluant les valeurs manquantes et qui indique le nombre de données manquantes par variable[4]

```
summary (n250)
```

```
##       num              sexe          cours1          cours2
##  Min.   :  1.00   Min.   :1.000   Min.   :-7.000   Min.   : 5.00
##  1st Qu.: 63.25   1st Qu.:2.000   1st Qu.: 9.000   1st Qu.:10.00
##  Median :125.50   Median :2.000   Median :10.000   Median :12.00
##  Mean   :125.50   Mean   :1.812   Mean   : 9.996   Mean   :12.02
##  3rd Qu.:187.75   3rd Qu.:2.000   3rd Qu.:11.000   3rd Qu.:14.00
##  Max.   :250.00   Max.   :2.000   Max.   :16.000   Max.   :20.00
##                                   NA's   :10       NA's   :13
##      cours3          cours4          cours5          cours6
##  Min.   :-6.00   Min.   : 0.000   Min.   : 4.00   Min.   : 2.00
##  1st Qu.: 9.00   1st Qu.: 7.000   1st Qu.: 9.00   1st Qu.: 8.00
##  Median :11.00   Median : 9.000   Median :10.00   Median :11.00
##  Mean   :11.08   Mean   : 8.946   Mean   :10.17   Mean   :10.85
##  3rd Qu.:14.00   3rd Qu.:11.000   3rd Qu.:12.00   3rd Qu.:13.00
##  Max.   :20.00   Max.   :18.000   Max.   :14.00   Max.   :19.00
##  NA's   :14      NA's   :11       NA's   :132     NA's   :127
```

Si la fonction `summary` s'accomode facilement des données manquantes, en n'incluant dans les calculs que les observations comportant des données non manquantes, il n'en n'est pas de même pour la plupart des autres fonctions statistiques de `R` qui requièrent d'indiquer explicitement le traitement à appliquer aux données manquantes. Ainsi, la fonction `mean` appliquée à la variable "cours1" du tableau de données ne calcule pas la moyenne :

```
mean (n250$cours1)
```

```
## [1] NA
```

L'argument `na.rm`[5] de la fonction `mean` permet d'indiquer que le calcul de la moyenne doit être fait en enlevant les données manquantes :

```
mean (n250$cours1, na.rm = TRUE)
```

```
## [1] 9.995833
```

[4] un examen attentif de ces résultats permet de relever des notes minimales étranges pour "cours1" et "cours3". Ce sont des erreurs de saisie introduites volontairement dans le tableau de données. Leur repérage et leur correction feront l'objet d'un exercice en travaux dirigés.

[5] "rm" pour *remove*.

5.2.3 L'indexation en présence de données manquantes

Pour extraire d'un tableau de données un sous-ensemble correspondant à une certaine condition, on a vu que l'on pouvait poser une contrainte sur les lignes. Par exemple, la commande `n250 [n250$sexe == 1, ]` permet d'extraire tous les hommes. Il est donc tentant, pour extraire tous les étudiants ayant une donnée manquante à "cours1" d'écrire : `n250 [n250$cours == -9, ]` ou `n250 [n250$cours == "NA", ]`. Ces commandes ne donneront pas les résultats escomptés.

Pour extraire les observations qui contiennent une donnée manquante, il faut utiliser une fonction spécifique, la fonction `is.na`, comme dans l'exemple suivant qui extrait du tableau de données tous les étudiants n'ayant pas de note à "cours1" :

```
n250 [is.na (n250$cours1), ]
```

```
##     num sexe cours1 cours2 cours3 cours4 cours5 cours6
## 2     2    1     NA     14     13     13     NA     NA
## 22   22    2     NA     13     13     11     NA     14
## 61   61    2     NA     15     16     12     10     NA
## 162 162    2     NA     11      9      8     NA     10
## 165 165    2     NA     10     16      6      8     NA
## 178 178    1     NA     14     10      6     NA     12
## 205 205    2     NA     17     20      3      9     NA
## 224 224    2     NA      6     NA     10     NA      9
## 245 245    2     NA     NA      4      9      6     NA
## 249 249    2     NA     13      8      9      7     NA
```

Une autre difficulté émerge lorsque l'on veut extraire un sous-ensemble en posant une contrainte (numérique par exemple) sur une variable qui comporte des données manquantes. Par exemple, si on souhaite extraire tous les étudiants qui ont une note supérieure à 14 à "cours1", on obtient une extraction un peu étonnante :

```
n250 [n250$cours1 > 14, ]
```

```
##      num sexe cours1 cours2 cours3 cours4 cours5 cours6
## NA    NA   NA     NA     NA     NA     NA     NA     NA
## NA.1  NA   NA     NA     NA     NA     NA     NA     NA
## NA.2  NA   NA     NA     NA     NA     NA     NA     NA
## 75    75    2     15     16     14      5      8     NA
## NA.3  NA   NA     NA     NA     NA     NA     NA     NA
## NA.4  NA   NA     NA     NA     NA     NA     NA     NA
## NA.5  NA   NA     NA     NA     NA     NA     NA     NA
## NA.6  NA   NA     NA     NA     NA     NA     NA     NA
## 214  214    2     15     18     18     12     NA     18
## NA.7  NA   NA     NA     NA     NA     NA     NA     NA
## 244  244    2     16     19     16      7     NA      7
## NA.8  NA   NA     NA     NA     NA     NA     NA     NA
## NA.9  NA   NA     NA     NA     NA     NA     NA     NA
```

Les étudiants extraits sont non seulement les trois qui ont une note supérieure à 14 à "cours1", mais aussi les 10 qui n'ont pas de note à "cours1". De plus, ces 10 derniers étudiants se voient affubler de codes `NA` à toutes les variables, même si ce n'est pas le cas. Nous ne rentrerons pas ici dans les explications détaillées de ce résultat un peu étrange à première vue[6]. Il existe toutefois plusieurs manières simples de contourner la difficulté, qui fournissent toutes le même résultat :

- Déclarer explicitement que les valeurs manquantes ne doivent pas être extraites[7] :

[6] En bref, c'est parce que la comparaison d'une valeur à une valeur manquante (`NA`) ne renvoie pas à une valeur logique (`TRUE` ou `FALSE`), mais à un code de donnée manquante (`NA`).

[7] Rappelons que le symbole `!` correspond à la négation logique.

```
n250 [n250$cours1 > 14 & !is.na (n250$cours1), ]
```

```
##     num sexe cours1 cours2 cours3 cours4 cours5 cours6
## 75   75    2     15     16     14      5      8     NA
## 214 214    2     15     18     18     12     NA     18
## 244 244    2     16     19     16      7     NA      7
```

- utiliser la fonction `which` :

```
n250 [which (n250$cours1 > 14), ]
```

```
##     num sexe cours1 cours2 cours3 cours4 cours5 cours6
## 75   75    2     15     16     14      5      8     NA
## 214 214    2     15     18     18     12     NA     18
## 244 244    2     16     19     16      7     NA      7
```

- utiliser la fonction `subset` :

```
subset (n250, n250$cours1 > 14)
```

```
##     num sexe cours1 cours2 cours3 cours4 cours5 cours6
## 75   75    2     15     16     14      5      8     NA
## 214 214    2     15     18     18     12     NA     18
## 244 244    2     16     19     16      7     NA      7
```

5.3 Une présentation systématique de l'extraction

Après une présentation globale des grands principes de l'indexation, l'objectif de cette section est de revenir de manière plus systématique sur les principales manières d'extraire des sous-ensembles (de lignes et/ou de colonnes) d'un tableau de données. Ces différentes méthodes sont plus ou moins efficaces et utiles selon les situations. C'est à l'utilisateur de décider de l'approche qui lui paraît la plus pertinente ou la plus facile à mettre en oeuvre.

5.3.1 Sélectionner des variables (les colonnes)

5.3.1.1 Sélectionner les variables par les numéros de colonnes

Les deux commandes suivantes donnent *a priori* le même résultat :

- `n250[3]`
- `n250[,3]`

La différence est subtile :

- avec la première commande[8], le résultat apparaît affiché en colonne et l'objet qui est créé est un tableau de données (`data frame`) ne comportant qu'une seule variable. Pour s'en convaincre, on n'affiche que les 6 premières valeurs :

```
head (n250[3])
```

```
##   cours1
## 1     11
## 2     NA
## 3      9
```

[8] On notera que, lors d'une indexation d'un tableau de données, si on n'indique aucun indice pour les lignes, `R` considère par défaut que l'on s'intéresse aux colonnes.

```
## 4      9
## 5     10
## 6     10
```

- avec la seconde commande, le résultat apparaît affiché en ligne et l'objet qui est créé est un vecteur :

```
head (n250[,3])
```

```
## [1] 11 NA  9  9 10 10
```

Plutôt que de sélectionner les colonnes souhaitées, on peut éliminer les colonnes non souhaitées. Les trois commandes suivantes donnent le même résultat, à savoir uniquement les colonnes 1 (numéro de l'étudiant) et 2 (son genre) du tableau de données :

- `n250[ , -c(3,4,5,6)]`
- `n250[ , -c(3:6)]`
- `n250[ , -(3:6)]`

Si on a besoin de faire référence souvent à plusieurs colonnes (variables), il peut être utile de stocker leurs numéros dans un vecteur :

```
cours <- 3:8
```

Il est alors très facile d'utiliser ce vecteur pour indexer le tableau de données. Dans l'exemple suivant, on extrait uniquement les notes des six premiers sujets (sans leur identifiant et leur sexe donc) :

```
n250 [1:6, cours]
```

```
##   cours1 cours2 cours3 cours4 cours5 cours6
## 1     11     13     11     15     13     NA
## 2     NA     14     13     13     NA     NA
## 3      9     12      9      9     11     NA
## 4      9     12     11     13     NA     11
## 5     10     10     NA     10      8     NA
## 6     10     12      9      8     NA      9
```

Evidemment, le symbole - fonctionne aussi :

```
n250 [1:6, -cours]
```

```
##   num sexe
## 1   1    2
## 2   2    1
## 3   3    2
## 4   4    2
## 5   5    1
## 6   6    2
```

La fonction `match` permet de retrouver facilement le numéro de colonne occupée par une variable, ce qui est particulièrement utile dans un grand tableau de données :

```
match ("cours1", names (n250))  # numéro de colonne de la variable cours1
```

```
## [1] 3
```

```
match ("cours6", names (n250))  # numéro de colonne de la variable cours6
```

```
## [1] 8
```

Lorsqu'on travaille avec les numéros de colonnes, la fonction `grep` est aussi très utile. Elle permet souvent d'économiser de longues et fastidieuses écritures. La commande suivante, par exemple, permet d'obtenir les

numéros de colonnes de toutes les variables comprenant la chaine de caractères "cours"[9] :

```
grep ("cours", names (n250))
```

```
## [1] 3 4 5 6 7 8
```

Et pour extraire du tableau les notes à tous les cours, on peut aussi écrire (ici pour les six premiers étudiants) :

```
n250 [1:6, grep ("cours", names (n250))]
```

```
##   cours1 cours2 cours3 cours4 cours5 cours6
## 1     11     13     11     15     13     NA
## 2     NA     14     13     13     NA     NA
## 3      9     12      9      9     11     NA
## 4      9     12     11     13     NA     11
## 5     10     10     NA     10      8     NA
## 6     10     12      9      8     NA      9
```

5.3.1.2 Sélectionner les variables par leurs noms

Les deux commandes suivantes fournissent le même résultat : un tableau de données avec uniquement les notes aux cours obligatoires :

- `n250[, c("cours1", "cours2", "cours3")]`
- `subset (n250, select = cours1:cours3)`

Si un ensemble de variables est utilisé souvent, il est très utile de stocker leurs noms dans un vecteur. Par exemple :

```
options <- c ("cours5", "cours6")
```

On peut alors utilise le vecteur "options" pour extraire les notes aux cours optionnels (des six premiers étudiants dans l'exemple qui suit) :

```
n250 [1:6, options]
```

```
##   cours5 cours6
## 1     13     NA
## 2     NA     NA
## 3     11     NA
## 4     NA     11
## 5      8     NA
## 6     NA      9
```

5.3.1.3 Sélectionner les variables par des vecteurs logiques

L'extraction de certaines colonnes du tableau peut aussi se faire par l'intermédiaires de valeurs logiques. Ainsi, les deux commandes suivantes permettent d'extraire le numéro de l'étudiant et sa note au cours3[10] :

- `n250[, c(TRUE, FALSE, TRUE, FALSE, FALSE, FALSE, FALSE, FALSE)]`
- `n250 [, as.logical (c(1,0,1,0,0,0,0,0))]`

[9] Pour bien comprendre la fonction `grep`, essayer par exemple : `grep ("u", names (n250))`.

[10] Evidemment, ce ne serait pas très judicieux d'utiliser ces commandes pour une extraction qui peut s'obtenir de manière beaucoup plus simple par d'autres moyens. Mais cet exemple permet d'introduire à l'utilisation d'indices logiques qui sont bien utiles – voire indispensables – dans d'autres circonstances.

Le vecteur logique peut être généré automatiquement avec la fonction `%in%`. Par exemple, le vecteur logique correspond à la sélection de « cours1 » et de « cours3 » s'obtient ainsi :

```
names (n250) %in% c ("cours1", "cours3")
```

```
## [1] FALSE FALSE  TRUE FALSE  TRUE FALSE FALSE FALSE
```

Et on peut donc sans difficulté utiliser :

```
n250 [1:6, names (n250) %in% c ("cours1", "cours3")]
```

```
##   cours1 cours3
## 1     11     11
## 2     NA     13
## 3      9      9
## 4      9     11
## 5     10     NA
## 6     10      9
```

5.3.1.4 Autres moyens

Nous avons déjà vu que le symbole `$` permet d'extraire une colonne particulière d'un tableau de données :

```
n250$cours1  # les notes de tous les individus à "cours1"
```

```
##   [1] 11 NA  9  9 10 10 11 13  7 10  8  9 13 13 14  9 10  9 11  8  8 NA  8
##  [24]  9 10  9  9  8  8  8  6 12 10 10 11  9 10 11  8  9 11 12  8  9  8 10
##  [47]  8 11 12  8 12  7  9 11  8  9 10 10  6 11 NA  8  9 11  8 10 11 12 14
##  [70]  7  6 11  8  7 15 10 10 10 10 13 10 11  9 11  9 12  9 12 10 13  9 12
##  [93] 10 10 11 14 12 11 11 11  8 12  6  8 11  9 11 12 11 13 13  8 11  9  9
## [116] 13 11  9  9  8 11  9 11 11 10 13 14  8 11  9  7 10 10  9  9 13 11 12
## [139]  8  8  7 13 10 10  9 10 11  8 10 12  4  7 10  9 10  9 11  8 11 11 11
## [162] NA 10  8 NA  7 10  9 14 10 11  9  8 11 12 11 11 NA 11 12  8 -7  8  9
## [185]  8  9 13  9 13 11 12  8 12 11  8 12 10  9 14  8  9  8 12  6 NA  9 13
## [208] 10  9 11 13 12  7 15 13 10 13  8 12 12 10 13 13 NA  8 10 11  9  9  9
## [231] 11 10  8 10 11 12 13 11  8 11 10  7 11 16 NA 11  8  9 NA  9
```

La fonction `attach` déclare un tableau de données référence et le nom des variables est alors accessible directement. Les commandes suivantes montrent que "cours1" n'est pas reconnu car cet objet n'est pas dans l'environnement de travail. Après avoir "attaché" le tableau de données, cette variable est accessible directement[11]

```
cours1
```

```
## Error in eval(expr, envir, enclos): objet 'cours1' introuvable
```

```
attach (n250)
cours1
```

```
##   [1] 11 NA  9  9 10 10 11 13  7 10  8  9 13 13 14  9 10  9 11  8  8 NA  8
##  [24]  9 10  9  9  8  8  8  6 12 10 10 11  9 10 11  8  9 11 12  8  9  8 10
##  [47]  8 11 12  8 12  7  9 11  8  9 10 10  6 11 NA  8  9 11  8 10 11 12 14
##  [70]  7  6 11  8  7 15 10 10 10 10 13 10 11  9 11  9 12  9 12 10 13  9 12
##  [93] 10 10 11 14 12 11 11 11  8 12  6  8 11  9 11 12 11 13 13  8 11  9  9
## [116] 13 11  9  9  8 11  9 11 11 10 13 14  8 11  9  7 10 10  9  9 13 11 12
## [139]  8  8  7 13 10 10  9 10 11  8 10 12  4  7 10  9 10  9 11  8 11 11 11
```

[11] Pour "détacher" un tableau, on utilise la fonction `detach` : `detach (n250)`

```
## [162] NA 10  8 NA  7 10  9 14 10 11  9  8 11 12 11 11 NA 11 12  8 -7  8  9
## [185]  8  9 13  9 13 11 12  8 12 11  8 12 10  9 14  8  9  8 12  6 NA  9 13
## [208] 10  9 11 13 12  7 15 13 10 13  8 12 12 10 13 13 NA  8 10 11  9  9  9
## [231] 11 10  8 10 11 12 13 11  8 11 10  7 11 16 NA 11  8  9 NA  9
```

Même si la fonction `attach` paraît très pratique, car elle évite de noter systématiquement le nom du tableau de données, nombre d'utilisateurs de `R` déconseillent son usage, car elle peut conduire à des confusions, notamment lorsque l'environnement de travail comporte des objets ayant des noms similaires à ceux des variables dans un tableau de données.

Les exemples précédents ont extrait les données du tableau sous forme de vecteurs. Il est donc logique que la commande suivante fournisse un seul vecteur avec les 250 valeurs de "cours2" qui viennent se positionner après les 250 valeurs de "cours1" :

```
c (n250$cours1, n250$cours2)
```

```
##   [1] 11 NA  9  9 10 10 11 13  7 10  8  9 13 13 14  9 10  9 11  8  8 NA  8
##  [24]  9 10  9  9  8  8  8  6 12 10 10 11  9 10 11  8  9 11 12  8  9  8 10
##  [47]  8 11 12  8 12  7  9 11  8  9 10 10  6 11 NA  8  9 11  8 10 11 12 14
##  [70]  7  6 11  8  7 15 10 10 10 10 13 10 11  9 11  9 12  9 12 10 13  9 12
##  [93] 10 10 11 14 12 11 11 11  8 12  6  8 11  9 11 12 11 13 13  8 11  9  9
## [116] 13 11  9  9  8 11  9 11 11 10 13 14  8 11  9  7 10 10  9  9 13 11 12
## [139]  8  8  7 13 10 10  9 10 11  8 10 12  4  7 10  9 10  9 11  8 11 11 11
## [162] NA 10  8 NA  7 10  9 14 10 11  9  8 11 12 11 11 NA 11 12  8 -7  8  9
## [185]  8  9 13  9 13 11 12  8 12 11  8 12 10  9 14  8  9  8 12  6 NA  9 13
## [208] 10  9 11 13 12  7 15 13 10 13  8 12 12 10 13 13 NA  8 10 11  9  9  9
## [231] 11 10  8 10 11 12 13 11  8 11 10  7 11 16 NA 11  8  9 NA  9 13 14 12
## [254] 12 10 12 13 10 13 11  9 11 15 12 15 12 10 13 13 11  9 13  5 10  8 15
## [277] 10 11 11 14  7 11  6 13 14  8 13 12 NA 12 12 14  8 13 10 18  8 15 NA
## [300]  8 19 NA 10 11  8  9 12 15 NA 13 15 12 11 11 NA 13 13 17 19  8  7 15
## [323]  9 10 16 14 12 13 14 12 14 15 11 14  7 15  9 15 14 10  9 13  8  8 15
## [346] 16 10 16 12 16 13 12 11 13 10 18 14 15 16 14 11 10 15 NA  9 14 14 11
## [369]  9  9 NA 12 11 12  8 15 18  8 14  8  7 12 11 10 11 20 14 14  8 10 11
## [392] 13 16 11 12 10 11  7 13 15  9 NA 11 14 12 12 13 13 15 18 14 11 15 NA
## [415] 10 10  9 11 19  8 17 14 11 15 13 14 14 14 NA 13  8  6 11 NA 16  9 13
## [438]  9 16 10 13 13 17 10  6 NA 12  9 12 11  6 11 16  9 17 15 12  8 12 14
## [461] 13 16  8 18 11 12 17 12 17 11 12  9 13  6  8 12 11 12 12  7 10 14 10
## [484] 13 13 16 11 13 11 14 12  5 13 19 NA 11  7  9 13 13
```

Pour obtenir un tableau avec « cours1 » et « cours2 » en colonnes, il faut les combiner dans un tableau avec la fonction `data.frame`:

```
head (data.frame (n250$cours1, n250$cours2))  # les six premières lignes seulement
```

```
##   n250.cours1 n250.cours2
## 1          11          13
## 2          NA          14
## 3           9          12
## 4           9          12
## 5          10          10
## 6          10          12
```

5.3.2 Sélectionner des observations (des lignes)

5.3.2.1 Sélectionner des observations par le numéro des lignes

La commande suivante permet de sélectionner les lignes 2, 4, 5 et 9 :

```
n250[c(2,4,5,9), ]
```

```
##   num sexe cours1 cours2 cours3 cours4 cours5 cours6
## 2   2    1     NA     14     13     13     NA     NA
## 4   4    2      9     12     11     13     NA     11
## 5   5    1     10     10     NA     10      8     NA
## 9   9    2      7     13      4      4     11     NA
```

5.3.2.2 Sélectionner des observations par le nom des lignes

Les lignes d'un tableau ont des noms par défaut[12] :

```
row.names (n250)
```

```
##   [1] "1"   "2"   "3"   "4"   "5"   "6"   "7"   "8"   "9"   "10"  "11"
##  [12] "12"  "13"  "14"  "15"  "16"  "17"  "18"  "19"  "20"  "21"  "22"
##  [23] "23"  "24"  "25"  "26"  "27"  "28"  "29"  "30"  "31"  "32"  "33"
##  [34] "34"  "35"  "36"  "37"  "38"  "39"  "40"  "41"  "42"  "43"  "44"
##  [45] "45"  "46"  "47"  "48"  "49"  "50"  "51"  "52"  "53"  "54"  "55"
##  [56] "56"  "57"  "58"  "59"  "60"  "61"  "62"  "63"  "64"  "65"  "66"
##  [67] "67"  "68"  "69"  "70"  "71"  "72"  "73"  "74"  "75"  "76"  "77"
##  [78] "78"  "79"  "80"  "81"  "82"  "83"  "84"  "85"  "86"  "87"  "88"
##  [89] "89"  "90"  "91"  "92"  "93"  "94"  "95"  "96"  "97"  "98"  "99"
## [100] "100" "101" "102" "103" "104" "105" "106" "107" "108" "109" "110"
## [111] "111" "112" "113" "114" "115" "116" "117" "118" "119" "120" "121"
## [122] "122" "123" "124" "125" "126" "127" "128" "129" "130" "131" "132"
## [133] "133" "134" "135" "136" "137" "138" "139" "140" "141" "142" "143"
## [144] "144" "145" "146" "147" "148" "149" "150" "151" "152" "153" "154"
## [155] "155" "156" "157" "158" "159" "160" "161" "162" "163" "164" "165"
## [166] "166" "167" "168" "169" "170" "171" "172" "173" "174" "175" "176"
## [177] "177" "178" "179" "180" "181" "182" "183" "184" "185" "186" "187"
## [188] "188" "189" "190" "191" "192" "193" "194" "195" "196" "197" "198"
## [199] "199" "200" "201" "202" "203" "204" "205" "206" "207" "208" "209"
## [210] "210" "211" "212" "213" "214" "215" "216" "217" "218" "219" "220"
## [221] "221" "222" "223" "224" "225" "226" "227" "228" "229" "230" "231"
## [232] "232" "233" "234" "235" "236" "237" "238" "239" "240" "241" "242"
## [243] "243" "244" "245" "246" "247" "248" "249" "250"
```

On peut donc sans difficulté (même si ce n'est pas très judicieux ici) sélectionner les lignes par leurs noms. Par exemple :

```
n250 [c("1", "3", "5"), ]
```

```
##   num sexe cours1 cours2 cours3 cours4 cours5 cours6
## 1   1    2     11     13     11     15     13     NA
## 3   3    2      9     12      9      9     11     NA
## 5   5    1     10     10     NA     10      8     NA
```

[12] Ces noms peuvent évidemment être changés. Essayer, par exemple : `row.names (n250) <- paste ("sujet num.", row.names (n250))`.

5.3.2.3 Sélectionner des observations par des vecteurs logiques

La commande suivante permet de créer un vecteur logique (`TRUE / FASLE`) en fonction de la valeur de vérité de la condition :

```
n250$sexe == 1
```

```
##   [1] FALSE  TRUE FALSE FALSE  TRUE FALSE  TRUE FALSE FALSE FALSE  TRUE
##  [12] FALSE FALSE  TRUE  TRUE FALSE FALSE  TRUE  TRUE FALSE FALSE FALSE
##  [23] FALSE FALSE FALSE  TRUE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE
##  [34]  TRUE  TRUE FALSE FALSE  TRUE FALSE  TRUE FALSE FALSE FALSE FALSE
##  [45] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [56] FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [67] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [78] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE
##  [89] FALSE  TRUE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [100]  TRUE FALSE FALSE FALSE FALSE  TRUE  TRUE FALSE FALSE FALSE FALSE
## [111]  TRUE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE
## [122]  TRUE FALSE  TRUE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE
## [133] FALSE FALSE FALSE  TRUE FALSE FALSE  TRUE FALSE  TRUE FALSE FALSE
## [144] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [155] FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [166] FALSE FALSE  TRUE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE
## [177] FALSE  TRUE FALSE FALSE FALSE  TRUE FALSE  TRUE FALSE FALSE FALSE
## [188] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [199] FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE
## [210]  TRUE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE
## [221] FALSE  TRUE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE  TRUE
## [232] FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE  TRUE
## [243] FALSE FALSE FALSE  TRUE  TRUE  TRUE FALSE FALSE
```

Tout naturellement donc, la commande suivante permettra d'extraire uniquement les hommes (on limite ici l'affichage aux 6 premières lignes avec la fonction `head`)[13] :

```
head (n250 [n250$sexe == 1, ])
```

```
##    num sexe cours1 cours2 cours3 cours4 cours5 cours6
## 2    2    1     NA     14     13     13     NA     NA
## 5    5    1     10     10     NA     10      8     NA
## 7    7    1     11     13      9     10     14     NA
## 11  11    1      8      9     NA     10     12     NA
## 14  14    1     13     12      9      9     NA      8
## 15  15    1     14     15     10     14     14     NA
```

Si la commande `n250$sexe == 1` retourne un vecteur logique, la fonction `which` permet de repérer, dans un vecteur, les *positions* des valeurs logiques `TRUE`. Ainsi, la commande suivante fournit les numéros de ligne des sujets de sexe masculin dans le tableau de données :

```
which (n250$sexe == 1)
```

```
##  [1]   2   5   7  11  14  15  18  19  26  29  34  35  38  40  58  87  90
## [18]  91 100 105 106 111 117 122 124 129 136 139 141 157 168 173 178 182
## [35] 184 201 209 210 215 222 226 231 236 242 246 247 248
```

[13] On notera que la commande `n250 [sexe == 1, ]` ne fonctionne pas, parce que l'objet sexe n'existe qu'à l'intérieur du tableau. Attention toutefois si un autre objet appelé `sexe` existe dans l'environnement de travail à l'extérieur du tableau de données : l'extraction produite risque d'être totalement fausse !).

Et on pourrait donc extraire les sujets masculins avec leurs numéros de ligne avec : `n250 [which (n250$sexe == 1), ]`

5.3.2.4 La fonction `subset`

La fonction `subset` est une alternative intéressante pour sélectionner les numéros de lignes. Elle permet notamment d'éviter les problèmes que l'on a vu plus haut en présence de données manquantes.

Voici quelques exemples qui peuvent se passer de commentaires :

```
subset (n250, sexe == 1)  # sélection de tous les individus masculins
```

```
##     num sexe cours1 cours2 cours3 cours4 cours5 cours6
## 2     2    1     NA     14     13     13     NA     NA
## 5     5    1     10     10     NA     10      8     NA
## 7     7    1     11     13      9     10     14     NA
## 11   11    1      8      9     NA     10     12     NA
## 14   14    1     13     12      9      9     NA      8
## 15   15    1     14     15     10     14     14     NA
## 18   18    1      9     13     16     11     NA      5
## 19   19    1     11     13     11      7     10     NA
## 26   26    1      9     15      9     11     NA      9
## 29   29    1      8     11      7     10      7     NA
## 34   34    1     10     13      6      9     NA     14
## 35   35    1     11     14     17      6     11     NA
## 38   38    1     11     12      6      9     NA     13
## 40   40    1      9     12     NA     11     NA     16
## 58   58    1     10     15     11     11     NA     12
## 87   87    1      9      9      9      9     10     NA
## 90   90    1     13     10      9      9     NA     10
## 91   91    1      9      9      8      9      5     NA
## 100 100    1     11     16     12     14     NA      9
## 105 105    1     11     10     17     12     12     NA
## 106 106    1      9     18     19     11     NA      6
## 111 111    1     13     11     16     10     11     NA
## 117 117    1     11     14      9     NA      9     NA
## 122 122    1      9     12      9     11     NA     11
## 124 124    1     11     12     11     15     NA     16
## 129 129    1     11     14     10     10     13     NA
## 136 136    1     13     20     19      7     NA     13
## 139 139    1      8      8      2      4      8     NA
## 141 141    1      7     11     11      8      8     NA
## 157 157    1     11     13     NA     11     12     NA
## 168 168    1      9     11     10      3     NA      9
## 173 173    1      8     11     11      8      9     10
## 178 178    1     NA     14     10      6     NA     12
## 182 182    1     -7      6     10      8     NA     10
## 184 184    1      9     NA     10     16     NA     16
## 201 201    1      9      6      7      8     12     NA
## 209 209    1      9     12      9      9     12     NA
## 210 210    1     11     14     10      4     NA      4
## 215 215    1     13     11      7     10     12     NA
## 222 222    1     13      9     13      3     NA     13
## 226 226    1     10     12     13      6     NA     10
## 231 231    1     11     10     14     16     11     NA
```

```
## 236 236    1     12     16     10     11     NA     12
## 242 242    1      7      5     17      8     NA     11
## 246 246    1     11     11     10     10     NA     11
## 247 247    1      8      7     15      9     14     NA
## 248 248    1      9      9     13     12     NA     15
```

```
subset (n250, cours1 > 14)  # sélection de tous les individus ayant plus de 14 à cours1
```

```
##     num sexe cours1 cours2 cours3 cours4 cours5 cours6
## 75   75    2     15     16     14      5      8     NA
## 214 214    2     15     18     18     12     NA     18
## 244 244    2     16     19     16      7     NA      7
```

```
# sélection de tous les individus ayant plus de 14 à cours1 ET plus de 10 à cours6
subset (n250, cours1 > 14 & cours6 > 10)
```

```
##     num sexe cours1 cours2 cours3 cours4 cours5 cours6
## 214 214    2     15     18     18     12     NA     18
```

Chapter 6

La manipulation des données

Le module de base de R comprend de nombreuses méthodes permettant de manipuler (i.e. recoder, calculer...) les variables et plus généralement les différents objets de l'environnement. Ce chapitre se limitera aux principales fonctions présentes dans ce noyau. Il existe toutefois un paquet qui connait beaucoup de succès parmi les utilisateurs de R et qui rend certaines manipulations plus simples, plus rapides et parfois aussi plus logiques pour un néophyte. Il s'agit du paquet `tidyr` (Wickham (2017) ; on pourra aussi consulter l'ouvrage de Wickham & Grolemund (2016) qui est accessible en ligne à cette adresse).

6.1 Remplacer (recoder) les valeurs d'une variable

6.1.1 Principes généraux

Il existe plusieurs manières pour remplacer les valeurs d'une variable par d'autres valeurs. Prenons la variable "age" du tableau de données `tab` qui contient les valeurs suivantes :

```
## [1] 68 25 34 28 45 29 54
```

Supposons que l'on s'aperçoive que l'âge du premier sujet n'est pas 68 ans, mais 86 ans. La manière la plus efficace de procéder à ce remplacement consiste à utiliser les "objets-logiques" de R. Un objet logique est un objet contenant des valeurs `TRUE` ou `FALSE`. Par exemple, la commande suivante va tester si chaque valeur du vecteur correspond à la condition logique indiquée ("est-ce que la valeur est égale à 68 ?"). Le résultat de cette commande fournit un vecteur de valeurs logiques : `TRUE` ou `FALSE`.

```
tab$age == 68
```

```
## [1]  TRUE FALSE FALSE FALSE FALSE FALSE FALSE
```

Si on combine cette condition logique avec l'indexation (cf. chapitre 5), il suffit d'indiquer par quelle valeur on veut remplacer celle qui est égale à "68".

```
tab$age[tab$age == 68] <- 86
tab$age
```

```
## [1] 86 25 34 28 45 29 54
```

La procédure est un peu plus compliquée si on souhaite, par exemple, changer le code "m" du sexe en code "h". Cela tient au fait que lors de la construction du tableau de données, la variable "sexe" est automatiquement transformée en un facteur et que R ne reconnaît pour ce facteur que les deux modalités de départ ("m" et "f")[1]. L'exécution de la commande `tab$sexe [tab$sexe == "m"] <- "h"` va donner lieu à un message

[1] La fonction `levels` permet de connaître les différentes modalités définies pour une variable de type facteur.

d'avertissement qui indique que la modalité `h` n'est pas une modalité valide et donc que toutes les valeurs `m` ont été recodées en données manquantes, ce qui est évidemment très ennuyeux. L'une des manières les plus simples de contourner ce problème est de créer une nouvelle variable (appellée "sexe2" dans l'exemple qui suit) définie avec les nouvelles modalités souhaitées.

```
tab$sexe2 [tab$sexe == "m"] <- "h"
tab$sexe2 [tab$sexe == "f"] <- "f"
tab
```

```
##   id sexe age revenu matrimo sexe2
## 1  1    f  86   1200       4     f
## 2  2    m  25   1100       1     h
## 3  3    f  34    850       2     f
## 4  4    f  28   2211       2     f
## 5  5    m  45   1210       3     h
## 6  6    f  29   1003       1     f
## 7  7    f  54  10110       1     f
```

La procédure indiquée ci-dessus pour recoder l'erreur commise sur l'âge (86 au lieu de 68) n'est possible que s'il n'y a qu'une seule personne dans le fichier qui a un âge de 68 ans. Si plusieurs valeurs dans la variable sont égales à 68, *toutes* ces valeurs seront remplacées par la nouvelle valeur ! Il est donc souvent plus prudent, de faire ce type de recodage sur la base de l'identifiant du sujet qui, lui, est unique dans le tableau de données[2].

Imaginons qu'il y ait eu une erreur de codage aussi sur la variable "matrimo" et que la personne #2 ne soit pas célibataire (code 1), mais vive maritalement (code 2). Il n'est pas possible d'utiliser la commande `tab$matrimo [tab$matrimo == 1] <- 2`, car cela reviendrait à recoder *toutes* les personnes célibataires.

La commande suivante permet d'extraire toutes les personnes déclarées célibataires et de repérer le numéro de la personne pour laquelle une erreur a été commise (la variable "id" est égale à 2).

```
tab [tab$matrimo == 1,]
```

```
##   id sexe age revenu matrimo sexe2
## 2  2    m  25   1100       1     h
## 6  6    f  29   1003       1     f
## 7  7    f  54  10110       1     f
```

Le recodage du sujet #2 peut alors se faire directement à partir de son numéro :

```
tab$matrimo [tab$id == 2] <- 2
tab
```

```
##   id sexe age revenu matrimo sexe2
## 1  1    f  86   1200       4     f
## 2  2    m  25   1100       2     h
## 3  3    f  34    850       2     f
## 4  4    f  28   2211       2     f
## 5  5    m  45   1210       3     h
## 6  6    f  29   1003       1     f
## 7  7    f  54  10110       1     f
```

6.1.2 Regrouper plusieurs modalités

Outre la correction d'erreurs de saisie, le recodage est utilisée fréquemment lorsque l'on souhaite regrouper plusieurs modalités. Imaginons que l'on souhaite créer une nouvelle variable dans le tableau où la situation

[2] D'où l'importance de veiller, lors de la constitution de son tableau de données, à ce que chaque personne ait un identifiant unique.

matrimoniale ne compte plus que deux catégories : les personnes qui vivent seules (code = 1) et les personnes qui vivent en couple (code = 2). On va d'abord créer une nouvelle variable "matrimo2" qui est une copie de la variable originale :

```
tab$matrimo2 <- tab$matrimo
tab
```

```
##   id sexe age revenu matrimo sexe2 matrimo2
## 1  1    f  86   1200       4     f        4
## 2  2    m  25   1100       2     h        2
## 3  3    f  34    850       2     f        2
## 4  4    f  28   2211       2     f        2
## 5  5    m  45   1210       3     h        3
## 6  6    f  29   1003       1     f        1
## 7  7    f  54  10110       1     f        1
```

Dans la nouvelle variable "matrimo2", le code "2" est correctement attribué (ce sont toutes les personnes qui vivent en couple). Il faut regrouper les codes "1", "3" et "4" dans le code "1". Cela peut se faire, par exemple, en utilisant le symbole `!=` (qui signifie "différent de")[3] :

```
tab$matrimo2[tab$matrimo2 != 2] <-  1
tab
```

```
##   id sexe age revenu matrimo sexe2 matrimo2
## 1  1    f  86   1200       4     f        1
## 2  2    m  25   1100       2     h        2
## 3  3    f  34    850       2     f        2
## 4  4    f  28   2211       2     f        2
## 5  5    m  45   1210       3     h        1
## 6  6    f  29   1003       1     f        1
## 7  7    f  54  10110       1     f        1
```

Pour recoder une variable quantitative en catégories, on peut utiliser le même principe. Par exemple, si on veut faire trois catégories d'âge dans une nouvelle variable appelée "age3", on écrira :

```
tab$age3 [tab$age >= 25 & tab$age <= 35] <- "moins de 35 ans"
tab$age3 [tab$age >= 36 & tab$age <= 45] <- "de 36 à 45 ans"
tab$age3 [tab$age >= 45] <- "plus de 45 ans"
tab
```

```
##   id sexe age revenu matrimo sexe2 matrimo2           age3
## 1  1    f  86   1200       4     f        1  plus de 45 ans
## 2  2    m  25   1100       2     h        2 moins de 35 ans
## 3  3    f  34    850       2     f        2 moins de 35 ans
## 4  4    f  28   2211       2     f        2 moins de 35 ans
## 5  5    m  45   1210       3     h        1  plus de 45 ans
## 6  6    f  29   1003       1     f        1 moins de 35 ans
## 7  7    f  54  10110       1     f        1  plus de 45 ans
```

Une possibilité plus ingénieuse (et plus rapide) est fournie par la fonction `cut` :

```
tab$age3bis <-  cut (tab$age, breaks = c(20,35,45,90))
tab
```

```
##   id sexe age revenu matrimo sexe2 matrimo2           age3 age3bis
## 1  1    f  86   1200       4     f        1  plus de 45 ans (45,90]
## 2  2    m  25   1100       2     h        2 moins de 35 ans (20,35]
```

[3] Il aurait été possible aussi d'utiliser la commande suivante (avec le symbole `|` qui correspond au "ou" logique), mais c'est plus long à écrire : `tab$matrimo2[tab$matrimo2 == 1 | tab$matrimo2 == 3 | tab$matrimo2 == 4] <- 1`.

```
## 3  3    f  34    850       2     f        2 moins de 35 ans (20,35]
## 4  4    f  28   2211       2     f        2 moins de 35 ans (20,35]
## 5  5    m  45   1210       3     h        1  plus de 45 ans (35,45]
## 6  6    f  29   1003       1     f        1 moins de 35 ans (20,35]
## 7  7    f  54  10110       1     f        1  plus de 45 ans (45,90]
```

Cette commande fournit une nouvelle variable (appelée "age3bis" ici) qui contient 3 modalités :

- `(20,35]` (i.e. de 20 à 34 ans) ;
- `(35,45]` (i.e. de 35 à 44 ans) ;
- `(45,90]` (i.e. de 45 à 89 ans)[4].

6.1.3 La fonction `ifelse`

Les transformations conditionnelles des variables peuvent aussi s'effectuer de manière très simple avec la fonction `ifelse`. La structure générale de cette fonction est la suivante :

```
ifelse (expression logique, ce qui est à faire si la condition logique est vraie, ce qui est à faire si la condition logique est fausse)
```

Les exemples suivants sur le tableau de données des notes aux examens permettront de comprendre son utilisation.

*Exemple 1 : créer une variable ("cours1r") dans le tableau distinguant les sujets qui ont réussi au "cours1" (**`réussite`**) et ceux qui ont échoué (**`échec`**)*

```
n250$cours1r <-  ifelse (n250$cours1 >= 10, "réussite", "échec")
head (n250)
```

```
##   num sexe cours1 cours2 cours3 cours4 cours5 cours6  cours1r
## 1   1    2     11     13     11     15     13     NA réussite
## 2   2    1     NA     14     13     13     NA     NA     <NA>
## 3   3    2      9     12      9      9     11     NA    échec
## 4   4    2      9     12     11     13     NA     11    échec
## 5   5    1     10     10     NA     10      8     NA réussite
## 6   6    2     10     12      9      8     NA      9 réussite
```

*Exemple 2 : mettre aussi dans la variable "cours1r" la modalité **`échec`** à ceux qui n'ont pas passé l'examen*

```
n250$cours1r <- ifelse (n250$cours1 >= 10 & ! is.na (n250$cours1), "réussite", "échec")
head (n250)
```

```
##   num sexe cours1 cours2 cours3 cours4 cours5 cours6  cours1r
## 1   1    2     11     13     11     15     13     NA réussite
## 2   2    1     NA     14     13     13     NA     NA    échec
## 3   3    2      9     12      9      9     11     NA    échec
## 4   4    2      9     12     11     13     NA     11    échec
## 5   5    1     10     10     NA     10      8     NA réussite
## 6   6    2     10     12      9      8     NA      9 réussite
```

*Exemple 3 : imbriquer plusieurs fonctions **`ifelse`** les unes dans les autres*

Pour créer une variable ("cours1r") avec les trois indications classiques (**`défaillant`**, **`ajourné`**, **`reçu`** on peut nicher les fonctions `ifelse` les unes dans les autres (mais attention aux parenthèses !)

[4]Dans les modalités construites de manière automatique, la parenthèse indique que la valeur est incluse et le crochet indique que la valeur est exclue.

```
n250$cours1r <-  ifelse (is.na (n250$cours1), "défaillant(e)",
                         ifelse (n250$cours1 >= 10, "reçu(e)", "ajourné(e)"))
head (n250)
```

```
##   num sexe cours1 cours2 cours3 cours4 cours5 cours6       cours1r
## 1   1    2     11     13     11     15     13     NA       reçu(e)
## 2   2    1     NA     14     13     13     NA     NA défaillant(e)
## 3   3    2      9     12      9      9     11     NA    ajourné(e)
## 4   4    2      9     12     11     13     NA     11    ajourné(e)
## 5   5    1     10     10     NA     10      8     NA       reçu(e)
## 6   6    2     10     12      9      8     NA      9       reçu(e)
```

Exemple 4 : La condition logique ne porte pas nécessairement sur la variable recodée

La fonction `ifelse` peut aussi être utilisée en posant une condition logique sur une autre variable que celle qui est recodée. L'exemple qui suit est totalement absurde (on ajoute un point à la note à "cours1" aux étudiants masculins et on retire un point aux autres) : il ne sert qu'à illustrer les possibilités de la fonction `ifelse` !

```
n250$cours1c <-  ifelse (n250$sexe == 1, n250$cours1 + 1, n250$cours1 - 1)
head (n250)
```

```
##   num sexe cours1 cours2 cours3 cours4 cours5 cours6       cours1r cours1c
## 1   1    2     11     13     11     15     13     NA       reçu(e)      10
## 2   2    1     NA     14     13     13     NA     NA défaillant(e)      NA
## 3   3    2      9     12      9      9     11     NA    ajourné(e)       8
## 4   4    2      9     12     11     13     NA     11    ajourné(e)       8
## 5   5    1     10     10     NA     10      8     NA       reçu(e)      11
## 6   6    2     10     12      9      8     NA      9       reçu(e)       9
```

6.2 Créer une nouvelle variable à partir d'une transformation mathématique

Pour créer une nouvelle variable à partir d'une transformation mathématique des variables originales et l'ajouter dans le tableau de données, on peut utiliser les opérateurs arithmétiques, comme dans l'exemple suivant où on crée une variable "revenufr" où le revenu est exprimé en francs.

```
tab$revenufr <-  tab$revenu * 6.56
tab
```

```
##   id sexe age revenu matrimo sexe2 matrimo2           age3 age3bis
## 1  1    f  86   1200       4     f        1  plus de 45 ans (45,90]
## 2  2    m  25   1100       2     h        2 moins de 35 ans (20,35]
## 3  3    f  34    850       2     f        2 moins de 35 ans (20,35]
## 4  4    f  28   2211       2     f        2 moins de 35 ans (20,35]
## 5  5    m  45   1210       3     h        1  plus de 45 ans (35,45]
## 6  6    f  29   1003       1     f        1 moins de 35 ans (20,35]
## 7  7    f  54  10110       1     f        1  plus de 45 ans (45,90]
##   revenufr
## 1  7872.00
## 2  7216.00
## 3  5576.00
## 4 14504.16
## 5  7937.60
```

```
## 6  6579.68
## 7 66321.60
```

```
mean (tab$revenufr)
```

```
## [1] 16572.43
```

Pour illustrer le fonctionnement d'autres fonctions permettant de créer de nouvelles variables à partir de la transformation mathématique d'autres variables, nous allons travailler provisoirement sur une partie du tableau n250 (que l'on appelle `extrait`), ne comprenant que les trois premiers cours et, surtout, uniquement les sujets qui ont des notes à ces trois enseignements[5]. Il y a 215 sujets qui vérifient cette contrainte :

```
extrait <-  n250 [!is.na (n250$cours1) & !is.na (n250$cours2) & !is.na (n250$cours3), 1:5]
nrow (extrait)  # nombre de lignes du nouveau tableau de donnéees
```

```
## [1] 215
```

Pour calculer la moyenne aux trois premiers cours (nouvelle variable "m1"), on peut utiliser :

```
extrait$m1 <-  (extrait$cours1 + extrait$cours2 + extrait$cours3)/3
head (extrait)
```

```
##   num sexe cours1 cours2 cours3        m1
## 1   1    2     11     13     11 11.666667
## 3   3    2      9     12      9 10.000000
## 4   4    2      9     12     11 10.666667
## 6   6    2     10     12      9 10.333333
## 7   7    1     11     13      9 11.000000
## 8   8    2     13     10      6  9.666667
```

Mais l'écriture de ce type de commande devient très vite très fastidieuse s'il y a beaucoup de variables. Il vaut mieux utiliser la fonction `apply`[6] qui permet d'appliquer une fonction sur un ensemble de lignes ou de colonnes d'un tableau.

```
extrait$m2 <-  apply (extrait[,3:5], 1, mean)
head (extrait)
```

```
##   num sexe cours1 cours2 cours3        m1        m2
## 1   1    2     11     13     11 11.666667 11.666667
## 3   3    2      9     12      9 10.000000 10.000000
## 4   4    2      9     12     11 10.666667 10.666667
## 6   6    2     10     12      9 10.333333 10.333333
## 7   7    1     11     13      9 11.000000 11.000000
## 8   8    2     13     10      6  9.666667  9.666667
```

Dans l'exemple ci-dessus, la fonction `apply` a trois arguments :

- le nom du tableau de données et les colonnes concernées
- un chiffre indiquant si les calculs doivent être effectuées par ligne (chiffre 1) ou par colonne (chiffre 2) ; ici, les calculs seront donc appliqués à chacune des lignes ;
- le nom de la fonction à appliquer : il s'agit de la moyenne dans l'exemple ci-dessous, mais bien d'autres fonctions sont possibles (`sum`, `log`, `sqrt`, ...)

La fonction `apply` peut donc aussi être utilisée pour effectuer des calculs en colonnes, comme l'illustre l'exemple ci-dessous[7] :

[5] Le cas, plus complexe, de la création de variables en présence de données manquantes sera abordé dans la section suivante.

[6] La fonction `apply` fait partie d'un ensemble de fonctions très puissantes qui peuvent automatiser des opérations sur des vecteurs, des tableaux de données ou des listes. Les principales sont `tapply`, `lapply` ou encore `sapply`.

[7] En l'occurrence, dans cet exemple, le recours à la fonction `apply` n'est pas vraiment justifié, car on obtient plus facilement les moyennes par colonnes avec des fonctions statistiques que l'on verra dans la seconde partie.

```
apply (extrait[,3:5], 2, mean)
```

```
##   cours1   cours2   cours3
## 10.09302 12.03721 11.13953
```

Pour des calculs arithmétiques simples, il existe des "raccourcis" à la fonction `apply`, comme `rowMeans`, `rowSums` (pour faire des calculs par ligne), voire `colMeans` ou `colSums` (pour faire des calculs par colonnes)[8]. Par exemple, pour calculer la moyenne des notes aux trois cours pour chaque sujet, on peut utiliser :

```
extrait$m3 <- rowMeans (extrait [, 3:5])
head (extrait)
```

```
##   num sexe cours1 cours2 cours3        m1        m2        m3
## 1   1    2     11     13     11 11.666667 11.666667 11.666667
## 3   3    2      9     12      9 10.000000 10.000000 10.000000
## 4   4    2      9     12     11 10.666667 10.666667 10.666667
## 6   6    2     10     12      9 10.333333 10.333333 10.333333
## 7   7    1     11     13      9 11.000000 11.000000 11.000000
## 8   8    2     13     10      6  9.666667  9.666667  9.666667
```

6.3 Automatiser le recodage de plusieurs variables

Dans un test ou un questionnaire comportant plusieurs dizaines de questions, il est fastidieux de recoder chaque variable une à une pour se conformer aux règles de correction. Dans le tout petit exemple suivant, il y a quatre questions de type QCM avec 4 modalités de réponse à chaque fois. Pour les questions #1 et #3, c'est la réponse 2 qui est la réponse correcte ; pour les questions #2 et #4, c'est la réponse 3.

```
qcm <-  data.frame (v1 = c(1,2,1,3,4,2,3,1,2), v2 = c(2,3,4,1,4,2,2,1,1),
                    v3 = c(2,3,1,1,1,2,3,3,1), v4 = c(3,4,2,2,2,3,4,4,2))
qcm
```

```
##   v1 v2 v3 v4
## 1  1  2  2  3
## 2  2  3  3  4
## 3  1  4  1  2
## 4  3  1  1  2
## 5  4  4  1  2
## 6  2  2  2  3
## 7  3  2  3  4
## 8  1  1  3  4
## 9  2  1  1  2
```

Comment faire pour coder ces variables de manière à exprimer la justesse (code 1) ou la fausseté de la réponse (code 0)) ? On peut gérer le recodage avec la fonction `ifelse` appliquée à chaque variable l'une après l'autre (exemple ci-dessous). Cela fonctionne, mais cela devient vite fastidieux dès lors que le nombre de variables augmente.

```
qcm$v1r <-  ifelse (qcm$v1 == 2, 1, 0)
qcm$v2r <-  ifelse (qcm$v2 == 3, 1, 0)
qcm$v3r <-  ifelse (qcm$v3 == 2, 1, 0)
qcm$v4r <-  ifelse (qcm$v4 == 3, 1, 0)
qcm
```

```
##   v1 v2 v3 v4 v1r v2r v3r v4r
```

[8] Remarquer les majuscules à l'initiale de "Means" ou de "Sums"

```
## 1  1  2  2  3   0   0   1   1
## 2  2  3  3  4   1   1   0   0
## 3  1  4  1  2   0   0   0   0
## 4  3  1  1  2   0   0   0   0
## 5  4  4  1  2   0   0   0   0
## 6  2  2  2  3   1   0   1   1
## 7  3  2  3  4   0   0   0   0
## 8  1  1  3  4   0   0   0   0
## 9  2  1  1  2   1   0   0   0
```

Une solution simple consiste alors à appliquer la fonction `ifelse` non pas sur chaque variable, mais sur chaque sous-ensemble de variables ayant la même bonne réponse. Dans ce petit exemple, cela pourra se faire ainsi :

```
qcm[,c("v1", "v3")] <- ifelse (qcm[,c("v1", "v3")] == 2, 1,0)
qcm[,c("v2", "v4")] <- ifelse (qcm[,c("v2", "v4")] == 3, 1,0)
qcm
```

```
##   v1 v2 v3 v4 v1r v2r v3r v4r
## 1  0  0  1  1   0   0   1   1
## 2  1  1  0  0   1   1   0   0
## 3  0  0  0  0   0   0   0   0
## 4  0  0  0  0   0   0   0   0
## 5  0  0  0  0   0   0   0   0
## 6  1  0  1  1   1   0   1   1
## 7  0  0  0  0   0   0   0   0
## 8  0  0  0  0   0   0   0   0
## 9  1  0  0  0   1   0   0   0
```

Ainsi, avec un QCM comportant quatre modalités de réponses, 4 lignes de commandes suffisent pour recoder tous les items, quel que soit leur nombre.

Plutôt que d'indiquer les *noms* des sous-ensembles de variables, on se rappelle qu'on peut indiquer leurs numéros de colonnes. Lorsque le tableau de données comporte beaucoup de variables, il n'est pas toujours aisé de repérer visuellement le numéro de colonne occupé par telle variable. Le recours à la fonction `match` peut alors s'avérer utile. Par exemple, pour savoir quel est le numéro de colonne occupée, dans le tableau `qcm`, par la variable "v1r" :

```
match ("v1r", names (qcm))
```

```
## [1] 5
```

6.4 Gestion des données manquantes

6.4.1 Les calculs statistiques avec des données manquantes

Dans l'exemple suivant, la première commande de calcul de moyenne pour "cours6" (dans le tableau de données `n250`) ne fournit aucun résultat parce que le logiciel `R` exige que l'utilisateur indique explicitement la manière dont les données manquantes doivent être gérées. L'utilisation de l'argument `na.rm = TRUE`[9] permet de calculer la moyenne sur les données disponibles (et donc de ne pas tenir compte des personnes ayant des données manquantes).

```
mean (n250$cours6)
```

```
## [1] NA
```

[9] `rm` comme abréviation de "*remove*".

```
mean (n250$cours6, na.rm = TRUE)
```

```
## [1] 10.85366
```

6.4.2 Les manipulations de variables avec des données manquantes

Si on veut calculer pour chaque étudiant la moyenne aux trois premiers cours du tableau `n250`, on s'aperçoit qu'aucune moyenne n'est calculée pour les étudiants à qui il manque une note (variable "m1").

```
n250$m1 <-  rowMeans (n250[,3:5])
head (n250)
```

```
##   num sexe cours1 cours2 cours3 cours4 cours5 cours6       cours1r cours1c
## 1   1    2     11     13     11     15     13     NA       reçu(e)      10
## 2   2    1     NA     14     13     13     NA     NA défaillant(e)      NA
## 3   3    2      9     12      9      9     11     NA    ajourné(e)       8
## 4   4    2      9     12     11     13     NA     11    ajourné(e)       8
## 5   5    1     10     10     NA     10      8     NA       reçu(e)      11
## 6   6    2     10     12      9      8     NA      9       reçu(e)       9
##         m1
## 1 11.66667
## 2       NA
## 3 10.00000
## 4 10.66667
## 5       NA
## 6 10.33333
```

Cette fonctionnalité (utile si les modalités de contrôle des connaissances prévoient qu'une moyenne ne doit être calculée que pour les étudiants ayant été présents aux trois examens) peut aussi être modifiée avec l'argument `na.rm = T` (variable "m2") :

```
n250$m2 <-  rowMeans (n250[,3:5], na.rm = T)
head (n250)
```

```
##   num sexe cours1 cours2 cours3 cours4 cours5 cours6       cours1r cours1c
## 1   1    2     11     13     11     15     13     NA       reçu(e)      10
## 2   2    1     NA     14     13     13     NA     NA défaillant(e)      NA
## 3   3    2      9     12      9      9     11     NA    ajourné(e)       8
## 4   4    2      9     12     11     13     NA     11    ajourné(e)       8
## 5   5    1     10     10     NA     10      8     NA       reçu(e)      11
## 6   6    2     10     12      9      8     NA      9       reçu(e)       9
##         m1       m2
## 1 11.66667 11.66667
## 2       NA 13.50000
## 3 10.00000 10.00000
## 4 10.66667 10.66667
## 5       NA 10.00000
## 6 10.33333 10.33333
```

6.4.3 Nouvelles variables conditionnelles à la valeur d'autres variables

Imaginons, pour les besoins de ce paragraphe, que les cours entrant dans le calcul de la moyenne des hommes et des femmes ne soient pas les mêmes[10] : "cours1", "cours2" et "cours5" pour les femmes ; "cours1", "cours2"

[10] Supposition absurde, évidemment !

et “cours6” pour les hommes.

La méthode la plus simple consiste à utiliser, comme nous l’avons déjà vu, la fonction `ifelse` (variable “m3”) :

```
n250$m3 <- ifelse (n250$sexe == 2, rowMeans(n250[,c(3,4,7)], na.rm = T),
                   rowMeans(n250[,c(3,4,8)], na.rm = T))
head (n250)
```

```
##   num sexe cours1 cours2 cours3 cours4 cours5 cours6       cours1r cours1c
## 1   1    2     11     13     11     15     13     NA       reçu(e)      10
## 2   2    1     NA     14     13     13     NA     NA défaillant(e)      NA
## 3   3    2      9     12      9      9     11     NA    ajourné(e)       8
## 4   4    2      9     12     11     13     NA     11    ajourné(e)       8
## 5   5    1     10     10     NA     10      8     NA       reçu(e)      11
## 6   6    2     10     12      9      8     NA      9       reçu(e)       9
##         m1       m2       m3
## 1 11.66667 11.66667 12.33333
## 2       NA 13.50000 14.00000
## 3 10.00000 10.00000 10.66667
## 4 10.66667 10.66667 10.50000
## 5       NA 10.00000 10.00000
## 6 10.33333 10.33333 11.00000
```

Il est possible de procéder différemment, avec autant de lignes de commande qu’il y a de conditions (variable “m4”) :

```
n250$m4 [n250$sexe == 1] <- rowMeans(n250[n250$sexe == 1,c(3,4,8)], na.rm = T)
n250$m4 [n250$sexe == 2] <- rowMeans(n250[n250$sexe == 2,c(3,4,7)], na.rm = T)
head (n250)
```

```
##   num sexe cours1 cours2 cours3 cours4 cours5 cours6       cours1r cours1c
## 1   1    2     11     13     11     15     13     NA       reçu(e)      10
## 2   2    1     NA     14     13     13     NA     NA défaillant(e)      NA
## 3   3    2      9     12      9      9     11     NA    ajourné(e)       8
## 4   4    2      9     12     11     13     NA     11    ajourné(e)       8
## 5   5    1     10     10     NA     10      8     NA       reçu(e)      11
## 6   6    2     10     12      9      8     NA      9       reçu(e)       9
##         m1       m2       m3       m4
## 1 11.66667 11.66667 12.33333 12.33333
## 2       NA 13.50000 14.00000 14.00000
## 3 10.00000 10.00000 10.66667 10.66667
## 4 10.66667 10.66667 10.50000 10.50000
## 5       NA 10.00000 10.00000 10.00000
## 6 10.33333 10.33333 11.00000 11.00000
```

Dans les trois exemples précédents (variables “m2”, “m3”, “m4”), la moyenne est calculée même si les étudiants étaient absents à un ou plusieurs examens. C’est une situation très favorable aux étudiants, mais les règlements d’examen sont souvent beaucoup plus stricts. Pour qu’une moyenne ne soit calculée que pour les étudiants présents à tous les examens, il suffit d’enlever l’argument `na.rm = T`, comme on l’a vu dans la création de la variable “m1”. Mais comment faire si les modalités de contrôle des connaissances prévoient que la moyenne soit calculée à condition que l’étudiant soit présent à au moins deux examens sur les trois ?

On peut commencer à compter le nombre d’absences aux différents cours concernés par le calcul de la moyenne (variable “nbrabs”) :

```
n250$nbrabs <- ifelse (n250$sexe == 2, rowSums (is.na (n250[,c(3,4,7)])),
                       rowSums(is.na (n250[,c(3,4,8)])))
```

```
head (n250)
```

```
##   num sexe cours1 cours2 cours3 cours4 cours5 cours6       cours1r cours1c
## 1   1    2     11     13     11     15     13     NA       reçu(e)      10
## 2   2    1     NA     14     13     13     NA     NA défaillant(e)      NA
## 3   3    2      9     12      9      9     11     NA    ajourné(e)       8
## 4   4    2      9     12     11     13     NA     11    ajourné(e)       8
## 5   5    1     10     10     NA     10      8     NA       reçu(e)      11
## 6   6    2     10     12      9      8     NA      9       reçu(e)       9
##         m1       m2       m3       m4 nbrabs
## 1 11.66667 11.66667 12.33333 12.33333      0
## 2       NA 13.50000 14.00000 14.00000      2
## 3 10.00000 10.00000 10.66667 10.66667      0
## 4 10.66667 10.66667 10.50000 10.50000      1
## 5       NA 10.00000 10.00000 10.00000      1
## 6 10.33333 10.33333 11.00000 11.00000      1
```

Le calcul de la moyenne peut se faire alors en emboîtant les `ifelse`, mais cela devient vite assez compliqué. On peut, plus simplement indiquer que les sujets ayant plus de deux absences n'ont pas de moyenne (variable "m5") :

```
n250$m5 <-  ifelse (n250$nbrabs < 2, n250$m3, NA)
head (n250)
```

```
##   num sexe cours1 cours2 cours3 cours4 cours5 cours6       cours1r cours1c
## 1   1    2     11     13     11     15     13     NA       reçu(e)      10
## 2   2    1     NA     14     13     13     NA     NA défaillant(e)      NA
## 3   3    2      9     12      9      9     11     NA    ajourné(e)       8
## 4   4    2      9     12     11     13     NA     11    ajourné(e)       8
## 5   5    1     10     10     NA     10      8     NA       reçu(e)      11
## 6   6    2     10     12      9      8     NA      9       reçu(e)       9
##         m1       m2       m3       m4 nbrabs       m5
## 1 11.66667 11.66667 12.33333 12.33333      0 12.33333
## 2       NA 13.50000 14.00000 14.00000      2       NA
## 3 10.00000 10.00000 10.66667 10.66667      0 10.66667
## 4 10.66667 10.66667 10.50000 10.50000      1 10.50000
## 5       NA 10.00000 10.00000 10.00000      1 10.00000
## 6 10.33333 10.33333 11.00000 11.00000      1 11.00000
```

Seconde partie : Analyses statistiques

Chapter 7

Analyses univariées

Parfois improprement appelées "statistiques descriptives"[1], les analyses univariées permettent d'obtenir des indications comme la tendance centrale des réponses, leur dispersion ou encore la manière dont elles se répartissent en traitant chaque variable séparément.

7.1 Statistiques pour variables numériques

Par l'expression "variables numériques" ou "variables quantitatives", on désignera ici toutes les variables ayant au moins un niveau de mesure ordinale même si, *stricto sensu*, ces statistiques concernent avant tout les variables à partir d'un niveau de mesure d'intervalles.

7.1.1 Statistiques de base

La fonction `summary` fournit des statistiques sur toutes les variables du tableau de données :

```
summary (n250)
```

```
##       num              sexe           cours1           cours2
##  Min.   :  1.00   Min.   :1.000   Min.   :-8.000   Min.   :-8.00
##  1st Qu.: 63.25   1st Qu.:2.000   1st Qu.: 8.000   1st Qu.:10.00
##  Median :125.50   Median :2.000   Median :10.000   Median :12.00
##  Mean   :125.50   Mean   :1.812   Mean   : 9.276   Mean   :10.98
##  3rd Qu.:187.75   3rd Qu.:2.000   3rd Qu.:11.000   3rd Qu.:14.00
##  Max.   :250.00   Max.   :2.000   Max.   :16.000   Max.   :20.00
##      cours3          cours4         cours5          cours6
##  Min.   :-8.00   Min.   :-8.0   Min.   :-9.00   Min.   :-9.000
##  1st Qu.: 8.00   1st Qu.: 6.0   1st Qu.:-9.00   1st Qu.:-9.000
##  Median :11.00   Median : 9.0   Median :-8.00   Median :-8.000
##  Mean   :10.01   Mean   : 8.2   Mean   : 0.08   Mean   : 0.788
##  3rd Qu.:14.00   3rd Qu.:11.0   3rd Qu.:10.00   3rd Qu.:11.000
##  Max.   :20.00   Max.   :18.0   Max.   :14.00   Max.   :19.000
```

Ces résultats permettent d'observer que les notes comportent des valeurs négatives correspondant à des données manquantes qu'il faut déclarer. Si cela n'est pas fait les statistiques calculées (comme la moyenne) sont bien évidemment absurdes.

[1] La description statistique peut se faire sur une seule variable,sur deux variables ou sur plusieurs variables comme on le verra. Autrement dit, les statistiques descriptives ne se limitent pas aux statistiques univariées.

```
# déclaration comme données manquantes de toutes les valeurs inférieures à 0 pour tous les cours
n250 [,3:8] [n250 [,3:8] < 0] <- NA
summary (n250)
```

```
##       num              sexe          cours1          cours2
##  Min.   :  1.00   Min.   :1.000   Min.   : 4.00   Min.   : 5.00
##  1st Qu.: 63.25   1st Qu.:2.000   1st Qu.: 9.00   1st Qu.:10.00
##  Median :125.50   Median :2.000   Median :10.00   Median :12.00
##  Mean   :125.50   Mean   :1.812   Mean   :10.07   Mean   :12.02
##  3rd Qu.:187.75   3rd Qu.:2.000   3rd Qu.:11.00   3rd Qu.:14.00
##  Max.   :250.00   Max.   :2.000   Max.   :16.00   Max.   :20.00
##                                   NA's   :11      NA's   :13
##      cours3          cours4          cours5          cours6
##  Min.   : 0.00   Min.   : 0.000   Min.   : 4.00   Min.   : 2.00
##  1st Qu.: 9.00   1st Qu.: 7.000   1st Qu.: 9.00   1st Qu.: 8.00
##  Median :11.00   Median : 9.000   Median :10.00   Median :11.00
##  Mean   :11.15   Mean   : 8.946   Mean   :10.17   Mean   :10.85
##  3rd Qu.:14.00   3rd Qu.:11.000   3rd Qu.:12.00   3rd Qu.:13.00
##  Max.   :20.00   Max.   :18.000   Max.   :14.00   Max.   :19.00
##  NA's   :15      NA's   :11       NA's   :132     NA's   :127
```

La fonction `summary` fournit des statistiques différentes selon que les variables sont définies comme numériques ou comme facteurs. Ainsi, si on définit les variables "num" et "sexe" comme des facteurs, les statistiques proposées pour ces deux variables ne sont plus les mêmes (on obtient un affichage du nombre d'occurrences des six modalités les plus fréquentes).

```
n250$num <- factor (n250$num)
n250$sexe <- factor (n250$sexe)
summary (n250)
```

```
##       num      sexe        cours1          cours2          cours3
##  1      :  1   1: 47   Min.   : 4.00   Min.   : 5.00   Min.   : 0.00
##  2      :  1   2:203   1st Qu.: 9.00   1st Qu.:10.00   1st Qu.: 9.00
##  3      :  1           Median :10.00   Median :12.00   Median :11.00
##  4      :  1           Mean   :10.07   Mean   :12.02   Mean   :11.15
##  5      :  1           3rd Qu.:11.00   3rd Qu.:14.00   3rd Qu.:14.00
##  6      :  1           Max.   :16.00   Max.   :20.00   Max.   :20.00
##  (Other):244           NA's   :11      NA's   :13      NA's   :15
##      cours4          cours5          cours6
##  Min.   : 0.000   Min.   : 4.00   Min.   : 2.00
##  1st Qu.: 7.000   1st Qu.: 9.00   1st Qu.: 8.00
##  Median : 9.000   Median :10.00   Median :11.00
##  Mean   : 8.946   Mean   :10.17   Mean   :10.85
##  3rd Qu.:11.000   3rd Qu.:12.00   3rd Qu.:13.00
##  Max.   :18.000   Max.   :14.00   Max.   :19.00
##  NA's   :11       NA's   :132     NA's   :127
```

Des fonctions existent pour calculer séparément les différentes statistiques. Les principales fonctions de base sont illustrées ci-dessous :

```
mean (n250$cours1, na.rm = T)  # moyenne
```

```
## [1] 10.06695
```

```
sd (n250$cours1, na.rm = T)  # écart type
```

```
## [1] 1.952082
```

```
var (n250$cours1, na.rm = T)  # variance
```

```
## [1] 3.810626
```

```
median (n250$cours1, na.rm = T)  # médiane
```

```
## [1] 10
```

Ces fonctions peuvent être combinées avec la fonction `round` de manière à limiter le nombre de décimales :

```
round (sd(n250$cours1, na.rm = T), 2)  # arrondi à deux décimales
```

```
## [1] 1.95
```

La fonction `apply` permet d'obtenir simultanément des statistiques sur plusieurs variables du tableau :

```
apply (n250[,3:8], 2, mean, na.rm = T)
```

```
##    cours1    cours2    cours3    cours4    cours5    cours6
## 10.066946 12.016878 11.153191  8.945607 10.169492 10.853659
```

```
round (apply (n250[,3:8], 2, sd, na.rm = T), 2)
```

```
## cours1 cours2 cours3 cours4 cours5 cours6
##   1.95   2.97   3.95   3.03   2.18   3.73
```

7.1.2 Ecrire ses propres fonctions

Il n'existe pas, dans le module de base, de fonction spécifique pour le calcul de l'erreur-standard de la moyenne. Il est toutefois très simple de la calculer à partir de sa formule[2] ($se = \frac{s}{\sqrt{n}}$) :

```
sd (n250$cours1, na.rm = T) / sqrt (sum (table(n250$cours1)))
```

```
## [1] 0.1262697
```

Si on trouve cette formule trop fastidieuse à écrire et à retenir, `R` offre la possibilité d'écrire ses propres fonctions qui sont sauvegardées dans l'environnement de travail comme des objets et qui peuvent être utilisées à tout moment. Par exemple, pour créer une fonction appelée `se` qui calcule l'erreur-standard de la moyenne :

```
se <-  function (x) sd (x, na.rm = T) / sqrt (sum (table(x)))
se (n250$cours1)  # cette nouvelle fonction peut être appliquée à n'importe quelle variable
```

```
## [1] 0.1262697
```

```
round (apply (n250[,3:8], 2, se), 2)  # elle peut être combinée avec d'autres fonctions
```

```
## cours1 cours2 cours3 cours4 cours5 cours6
##   0.13   0.19   0.26   0.20   0.20   0.34
```

[2] Dans cette commande, on utilise `sum (table (n250$cours1))` pour obtenir le nombre total d'observations non manquantes pour "cours1".

De manière similaire, on peut créer une fonction un peu plus complexe qui calcule un intervalle de confiance autour de la moyenne. Cette nouvelle fonction s'appelle `IC` :

```
IC <- function (x, p = 0.95)
{
  se <- sd (x, na.rm = T) / sqrt (sum (table(x)))
  inf <- mean (x, na.rm = T) - qnorm((1-p)/2 + p) * se
  sup <- mean (x, na.rm = T) + qnorm((1-p)/2 + p) * se
  cat ("[", inf, ";", sup, "]", "(", p * 100, "%)", sep = "")
}
```

Par défaut, la fonction `IC` calcule un intervalle de confiance à 95% :

```
IC (n250$cours1)
```

```
## [9.819462;10.31443](95%)
```

On peut aussi indiquer quel est le pourcentage de confiance que l'on souhaite. L'exemple suivant fournit un intervalle de confiance à 90% :

```
IC (n250$cours1, p = 0.9)
```

```
## [9.85925;10.27464](90%)
```

Les fonctions créées le sont dans l'environnement de travail et "disparaissent" de cet environnement dès que l'on quitte le logiciel et qu'on le lance une nouvelle fois. Une petite manipulation permet toutefois de charger automatiquement ses propres fonctions dès qu'on ouvre le logiciel. Pour cela, il faut :

- Ecrire toutes les fonctions que l'on souhaite charger automatiquement dans un fichier de commandes que l'on appellera, par exemple : `fonctions.R` et noter soigneusement le chemin d'accès au répertoire dans lequel se trouve ce fichier (par exemple : `C:\Users\Jean Luc Kop\Documents\cours\lmd\M1 psycho\analyse quantitative\fonctions.R`).
- Ouvrir le fichier `Rpofile.site` (par exemple dans `RStudio`) qui se trouve dans le répertoire "etc" du dossier d'installation du logiciel R (le chemin d'accès ressemble à quelque chose comme : `C:\Program Files\R\R-3.4.1\etc`)[3].
- Ajouter dans le fichier `Rpofile.site` les commandes suivantes (en adaptant bien évidemment le chemin d'accès au fichier `fonctions.R`) :

```
.First <- function()
  {
    source ("C:/Users/Jean Luc Kop/Documents/cours/lmd/M1 psycho/analyse quantitative/fonctions.R")
  }
```

- Sauvegarder le fichier `Rpofile.site`.
- A chaque nouveau lancement de `R`, toutes les fonctions stockées dans le fichier `fonctions.R` seront ouvertes automatiquement dans l'environnement de travail.

7.1.3 Utiliser des fonctions dans des paquets

On trouve dans certains paquets des fonctions "prêtes à l'emploi" qui calculent directement plusieurs statistiques descriptives. C'est le cas par exemple de la fonction `describe` du paquet `prettyR` (Lemon & Grosjean (2015)).

```
library (prettyR)
```

[3]Les noms des dossiers de ce chemin sont bien sûr à adapter en fonction du système de chaque ordinateur et de la version de `R` utilisée (dans cet exemple, il s'agit de la version 3.4.1).

```
##
## Attachement du package : 'prettyR'

## The following objects are masked from 'package:psych':
##
##     describe, skew
```

```
describe (n250 [, 3:8])
```

```
## Description of n250[, 3:8]

##
##  Numeric
##         mean median   var   sd valid.n
## cours1 10.07     10  3.81 1.95     239
## cours2 12.02     12  8.81 2.97     237
## cours3 11.15     11 15.63 3.95     235
## cours4  8.95      9  9.19 3.03     239
## cours5 10.17     10  4.74 2.18     118
## cours6 10.85     11 13.95 3.73     123
```

Une liste plus complète de statistiques univariées (dont l'erreur standard de la moyenne) est produite par la fonction `describe` du paquet `psych` (Revelle (2017))[4]. :

```
library (psych)
psych:: describe (n250)
```

```
##        vars   n   mean    sd median trimmed   mad min max range  skew
## num*      1 250 125.50 72.31  125.5  125.50 92.66   1 250   249  0.00
## sexe*     2 250   1.81  0.39    2.0    1.89  0.00   1   2     1 -1.59
## cours1    3 239  10.07  1.95   10.0   10.02  1.48   4  16    12  0.16
## cours2    4 237  12.02  2.97   12.0   11.97  2.97   5  20    15  0.10
## cours3    5 235  11.15  3.95   11.0   11.17  4.45   0  20    20 -0.08
## cours4    6 239   8.95  3.03    9.0    8.93  2.97   0  18    18 -0.01
## cours5    7 118  10.17  2.18   10.0   10.24  2.22   4  14    10 -0.36
## cours6    8 123  10.85  3.73   11.0   10.80  4.45   2  19    17  0.11
##        kurtosis   se
## num*      -1.21 4.57
## sexe*      0.52 0.02
## cours1    -0.10 0.13
## cours2    -0.25 0.19
## cours3    -0.23 0.26
## cours4    -0.04 0.20
## cours5    -0.16 0.20
## cours6    -0.39 0.34
```

On notera qu'il arrive parfois, comme ici, que des fonctions aient le même nom dans des paquets différents. Pour éviter toute confusion, il est souhaitable d'ajouter dans ce cas le nom du paquet devant la fonction. Ainsi, `psych::describe (n250)` fournira le résultat de l'exécution de la fonction `describe` du paquet `psych` et `prettyR::describe (n250)` fournira le résultat de l'exécution de la fonction `describe` du paquet `prettyR`.

[4] L'auteur du paquet `psych`, William Revelle, a rédigé un très bon ouvrage d'introduction à la psychométrie et ses applications avec `R` qu'il met à disposition sur son site

7.2 Statistiques pour variables nominales (facteurs)

Des tableaux de fréquence plus élaborés que ceux issus de la fonction `summary` sont accessibles avec la fonction `table` et différents compléments. Dans l'exemple qui suit, ces fonctions sont illustrées avec la variable "cours1" qui est une variable numérique, mais des tableaux de fréquence ont aussi du sens sur ce type de variable.

```
# tableau de fréquence (nombre d'occurrences de chaque modalité)
table (n250$cours1)
```

```
##
##  4  6  7  8  9 10 11 12 13 14 15 16
##  1  5 10 39 44 40 48 23 20  6  2  1
```

```
sum (table (n250$cours1)) # nombre d'observations valides
```

```
## [1] 239
```

```
cumsum (table (n250$cours1)) # effectifs cumulés
```

```
##   4   6   7   8   9  10  11  12  13  14  15  16
##   1   6  16  55  99 139 187 210 230 236 238 239
```

```
table (n250$cours1) / sum (table (n250$cours1)) * 100 # pourcentages
```

```
##
##         4         6         7         8         9        10
##  0.4184100  2.0920502  4.1841004 16.3179916 18.4100418 16.7364017
##        11        12        13        14        15        16
## 20.0836820  9.6234310  8.3682008  2.5104603  0.8368201  0.4184100
```

```
cumsum (table (n250$cours1) / sum (table (n250$cours1)) * 100) # pourcentages cumulés
```

```
##         4         6         7         8         9        10
##   0.418410   2.510460   6.694561  23.012552  41.422594  58.158996
##        11        12        13        14        15        16
##  78.242678  87.866109  96.234310  98.744770  99.581590 100.000000
```

```
round (table (n250$cours1) / sum (table (n250$cours1)) * 100, 2) # pourcentages arrondis à deux décimal
```

```
##
##    4    6    7    8    9   10   11   12   13   14   15   16
##  0.42 2.09 4.18 16.32 18.41 16.74 20.08 9.62 8.37 2.51 0.84 0.42
```

```
round (cumsum (table (n250$cours1) / sum (table (n250$cours1)) * 100),2) # pourcentages cumulés arrondi
```

```
##     4     6     7     8     9    10    11    12    13    14
##  0.42  2.51  6.69 23.01 41.42 58.16 78.24 87.87 96.23 98.74
##    15    16
## 99.58 100.00
```

```
# tableau de fréquence (nombre d'occurrences de chaque modalité)
table (n250$cours1)
```

```
##
##  4  6  7  8  9 10 11 12 13 14 15 16
##  1  5 10 39 44 40 48 23 20  6  2  1
```

```
sum (table (n250$cours1)) # nombre d'observations valides
```

```
## [1] 239
```

```
cumsum (table (n250$cours1)) # effectifs cumulés
```

```
##   4   6   7   8   9  10  11  12  13  14  15  16
##   1   6  16  55  99 139 187 210 230 236 238 239
```

```
table (n250$cours1) / sum (table (n250$cours1)) * 100 # pourcentages
```

```
##
##          4          6          7          8          9         10
##  0.4184100  2.0920502  4.1841004 16.3179916 18.4100418 16.7364017
##         11         12         13         14         15         16
## 20.0836820  9.6234310  8.3682008  2.5104603  0.8368201  0.4184100
```

```
cumsum (table (n250$cours1) / sum (table (n250$cours1)) * 100) # pourcentages cumulés
```

```
##          4          6          7          8          9         10
##   0.418410   2.510460   6.694561  23.012552  41.422594  58.158996
##         11         12         13         14         15         16
##  78.242678  87.866109  96.234310  98.744770  99.581590 100.000000
```

```
round (table (n250$cours1) / sum (table (n250$cours1)) * 100, 2) # pourcentages arrondis à deux décimal
```

```
##
##     4     6     7     8     9    10    11    12    13    14    15    16
##  0.42  2.09  4.18 16.32 18.41 16.74 20.08  9.62  8.37  2.51  0.84  0.42
```

```
round (cumsum (table (n250$cours1) / sum (table (n250$cours1)) * 100),2) # pourcentages cumulés arrondi
```

```
##      4      6      7      8      9     10     11     12     13     14
##   0.42   2.51   6.69  23.01  41.42  58.16  78.24  87.87  96.23  98.74
##     15     16
##  99.58 100.00
```

```
# tableau de fréquence (nombre d'occurrences de chaque modalité)
table (n250$cours1)
```

```
##
##  4  6  7  8  9 10 11 12 13 14 15 16
##  1  5 10 39 44 40 48 23 20  6  2  1
```

```
# nombre d'observations valides
sum (table (n250$cours1))
```

```
## [1] 239
```

```
# effectifs cumulés
cumsum (table (n250$cours1))
```

```
##   4   6   7   8   9  10  11  12  13  14  15  16
##   1   6  16  55  99 139 187 210 230 236 238 239
```

```
# pourcentages
table (n250$cours1) / sum (table (n250$cours1)) * 100
```

```
##
##          4          6          7          8          9         10
##  0.4184100  2.0920502  4.1841004 16.3179916 18.4100418 16.7364017
##         11         12         13         14         15         16
## 20.0836820  9.6234310  8.3682008  2.5104603  0.8368201  0.4184100
```

```
# pourcentages cumulés
cumsum (table (n250$cours1) / sum (table (n250$cours1)) * 100)
```

```
##          4          6          7          8          9         10
##   0.418410   2.510460   6.694561  23.012552  41.422594  58.158996
##         11         12         13         14         15         16
##  78.242678  87.866109  96.234310  98.744770  99.581590 100.000000
```

```
# pourcentages cumulés arrondis à deux décimales
round (cumsum (table (n250$cours1) / sum (table (n250$cours1)) * 100),2)
```

```
##      4      6      7      8      9     10     11     12     13     14
##   0.42   2.51   6.69  23.01  41.42  58.16  78.24  87.87  96.23  98.74
##     15     16
##  99.58 100.00
```

Ici aussi, la fonction `apply` permet d'obtenir des analyses simultanées sur plusieurs variables :

```
# Tableaux de fréquences (occurrences) pour les variables cours1, cours2 et cours3
apply (n250[,3:5], 2, table)
```

```
## $cours1
##
##  4  6  7  8  9 10 11 12 13 14 15 16
##  1  5 10 39 44 40 48 23 20  6  2  1
##
## $cours2
##
##  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
##  2  5  7 18 17 21 32 31 33 25 19 11  6  5  4  1
##
## $cours3
##
##  0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
##  1  1  3  2  5  4 11 13 17 26 21 21 22 23 18 14 12  5  9  4  3
```

```
# pourcentages arrondis à une décimale pour les variables cours1, cours2 et cours3
apply (n250[,3:5], 2, function (x) round (table (x)/sum (table (x)) * 100, 1))
```

```
## $cours1
## x
##    4    6    7    8    9   10   11   12   13   14   15   16
##  0.4  2.1  4.2 16.3 18.4 16.7 20.1  9.6  8.4  2.5  0.8  0.4
##
## $cours2
## x
##    5    6    7    8    9   10   11   12   13   14   15   16   17   18   19
```

```
##  0.8  2.1  3.0  7.6  7.2  8.9 13.5 13.1 13.9 10.5  8.0  4.6  2.5  2.1  1.7
##   20
##  0.4
##
## $cours3
## x
##    0    1    2    3    4    5    6    7    8    9   10   11   12   13   14
##  0.4  0.4  1.3  0.9  2.1  1.7  4.7  5.5  7.2 11.1  8.9  8.9  9.4  9.8  7.7
##   15   16   17   18   19   20
##  6.0  5.1  2.1  3.8  1.7  1.3
```

Pour disposer de tableaux de fréquences complets, on peut rédiger une fois pour toute sa propre fonction (appelée `tabfreq` ici) :

```
tabfreq <- function(x)
{
t <- table (x, useNA = "always")
d1 <- cbind (effectif = t,
             "effectif cumulé" = cumsum (t),
             pourcentages = round (t/sum (t) * 100, 2),
             "pourcentages cumulés" = round (cumsum (t / sum (t) * 100), 2))
t <- table (x)
d2 <- cbind (effectif = t,
             "effectif cumulé" = cumsum (t),
             pourcentages = round (t/sum (t) * 100, 2),
             "pourcentages cumulés" = round (cumsum (t / sum (t) * 100), 2))
cat ("Avec toutes les observations", "\n")
print (d1)
cat ("\n")
cat ("Avec les observations valides uniquement", "\n")
print (d2)
}
```

Cette fonction permet d'obtenir deux tableaux : le premier établi à partir de toutes les observations (données manquantes incluses) et le second établi uniquement à partir des données valides (données manquantes exclues) :

```
tabfreq(n250$cours5)
```

```
## Avec toutes les observations
##      effectif effectif cumulé pourcentages pourcentages cumulés
## 4           1               1         0.4                  0.4
## 5           3               4         1.2                  1.6
## 6           2               6         0.8                  2.4
## 7           7              13         2.8                  5.2
## 8          11              24         4.4                  9.6
## 9          21              45         8.4                 18.0
## 10         17              62         6.8                 24.8
## 11         21              83         8.4                 33.2
## 12         21             104         8.4                 41.6
## 13          6             110         2.4                 44.0
## 14          8             118         3.2                 47.2
## <NA>      132             250        52.8                100.0
##
## Avec les observations valides uniquement
##    effectif effectif cumulé pourcentages pourcentages cumulés
```

```
## 4         1               1         0.85               0.85
## 5         3               4         2.54               3.39
## 6         2               6         1.69               5.08
## 7         7              13         5.93              11.02
## 8        11              24         9.32              20.34
## 9        21              45        17.80              38.14
## 10       17              62        14.41              52.54
## 11       21              83        17.80              70.34
## 12       21             104        17.80              88.14
## 13        6             110         5.08              93.22
## 14        8             118         6.78             100.00
```

Si on n'utilise pas sa propre fonction, la fonction `freq` du paquet `prettyR` peut être utile (mais elle ne permet pas d'avoir les pourcentages cumulés)[5] :

```
freq (n250$cours5, decr.order = F)
```

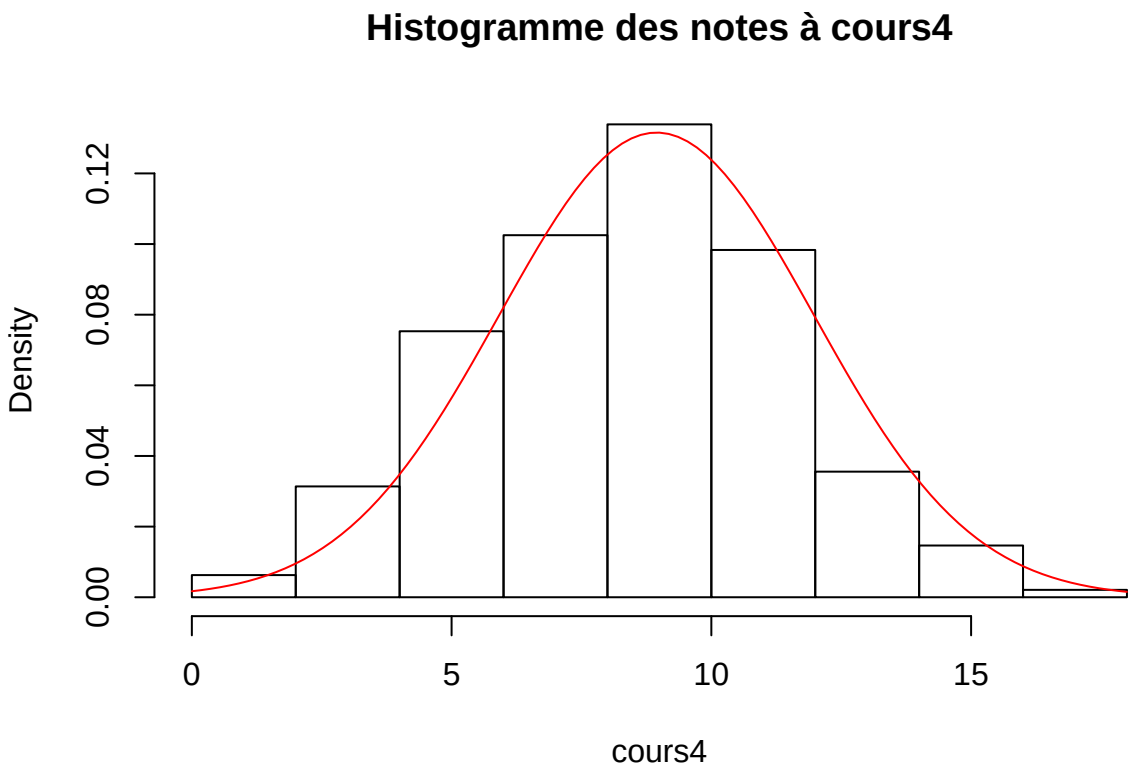
```
##
## Frequencies for n250$cours5
##         4    5    6    7    8    9   10   11   12   13   14   NA
##         1    3    2    7   11   21   17   21   21    6    8  132
## %     0.4  1.2  0.8  2.8  4.4  8.4  6.8  8.4  8.4  2.4  3.2 52.8
## %!NA  0.8  2.5  1.7  5.9  9.3 17.8 14.4 17.8 17.8  5.1  6.8
```

7.3 Représentations graphiques

R offre de très nombreuses possibilités graphiques et beaucoup de souplesse pour gérer les différents paramètres (titres, légendes, ajout de textes...). On se contentera ici de donner un exemple pour superposer la courbe d'une distribution normale à un histogramme :

```
hist(n250$cours4, prob=T, main = "Histogramme des notes à cours4", xlab = "cours4")
curve(dnorm (x, mean=mean(n250$cours4, na.rm = T),
            sd=sd(n250$cours4, na.rm = T)), col=2, add = TRUE)
```

[5] Dans cette fonction, l'argument `decr.order = F` permet d'avoir la liste des modalités dans l'ordre numérique (par défaut, les modalités les plus fréquentes apparaissent en premier).

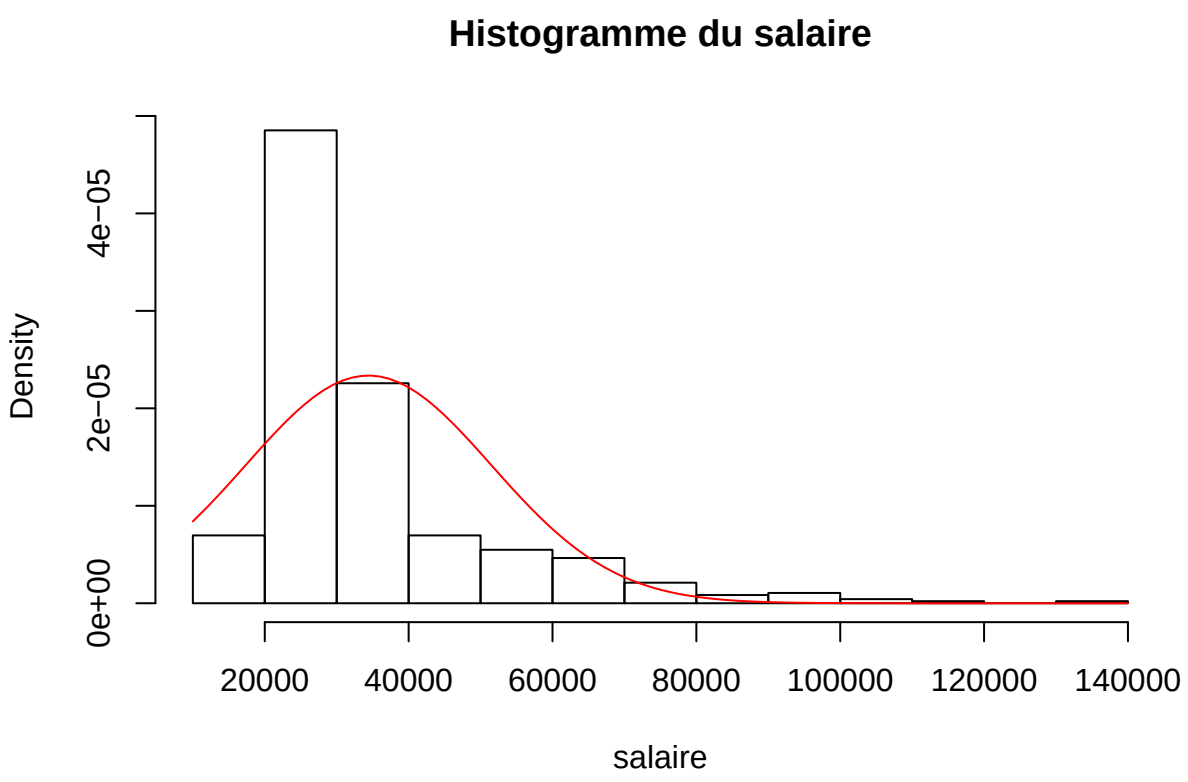


7.4 Transformation de la forme de distributions

Certaines variables ont des distributions assez éloignées d'une distribution normale, alors que ce postulat est requis pour nombre d'analyses statistiques. Les cas de distributions de revenus ou de temps de réponses sont des exemples classiques. On peut utiliser des transformations non linéaires (racine carrée, inverse, logarithme....) afin de les transformer et de se rapprocher d'une distribution normale comme l'illustre l'exemple ci-dessous.

Cette illustration utilise le tableau de données "employes.csv" qui peut être téléchargé en suivant ce lien. La variable "salact" est le salaire d'employés d'une entreprise (salaire brut annuel).

```
emp <- read.csv2("employes.csv")
hist(emp$salact, prob=T, main = "Histogramme du salaire", xlab = "salaire")
curve(dnorm (x, mean=mean(emp$salact, na.rm = T),
             sd=sd(emp$salact, na.rm = T)), col=2, add = TRUE)
```



La non-normalité de la distribution est confirmée par les valeurs élevées de l'indicateur d'asymétrie (*skewness*) et d'aplatissement (*kurtosis*) :

```
psych::describe (emp$salact)
```

```
##    vars   n     mean       sd median  trimmed     mad   min    max  range
## X1    1 474 34419.57 17075.66  28875 31199.14 8562.01 15750 135000 119250
##    skew kurtosis     se
## X1 2.11     5.27 784.31
```

Différentes transformations mathématiques peuvent être appliquées (inverse, logarithme et racine carrée) :

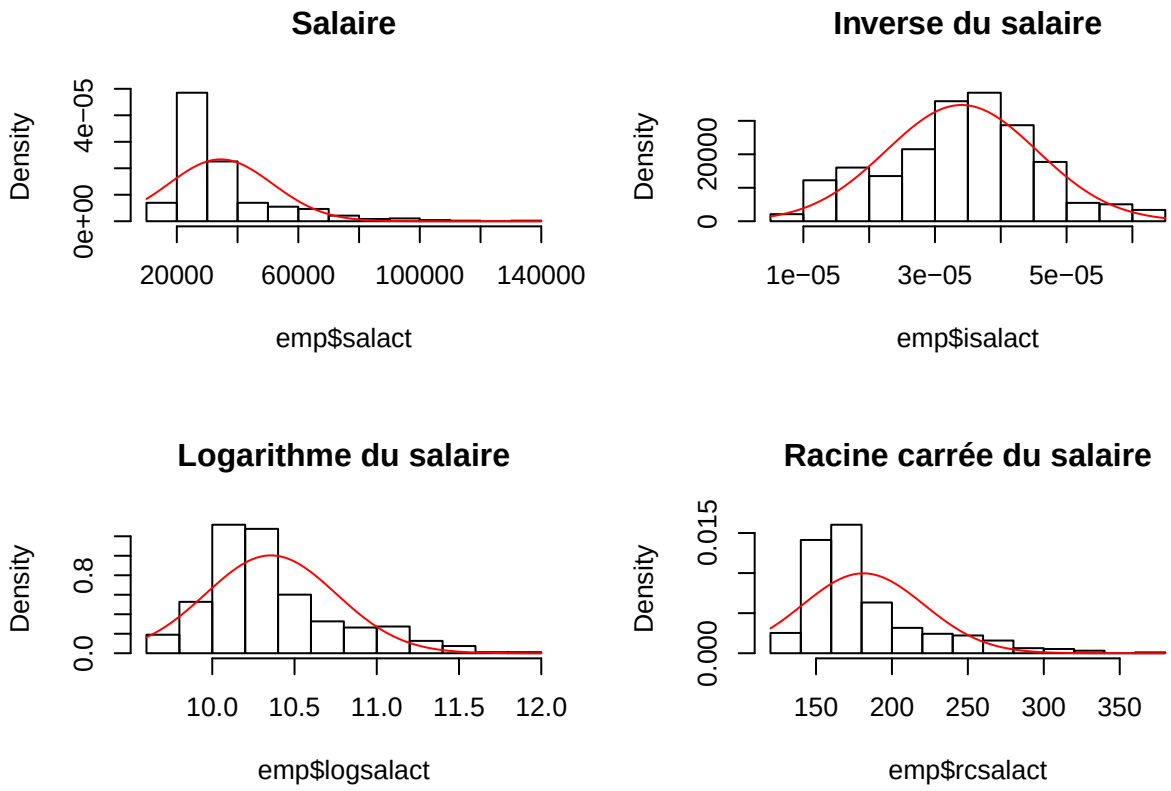
```
emp$isalact <- 1/emp$salact # inverse
emp$logsalact <- log (emp$salact) # logarithme
emp$rcsalact <- sqrt (emp$salact) # racine carrée
```

Les représentations graphiques montrent que la forme des distributions change de manière importante selon la transformation utilisée[6] :

```
par (mfrow = c(2,2))
hist(emp$salact, prob=T, main = "Salaire")
curve(dnorm (x, mean=mean(emp$salact, na.rm = T),
             sd=sd(emp$salact, na.rm = T)), col=2, add = TRUE)
hist(emp$isalact, prob=T, main = "Inverse du salaire")
curve(dnorm (x, mean=mean(emp$isalact, na.rm = T),
             sd=sd(emp$isalact, na.rm = T)), col=2, add = TRUE)
```

[6]Dans les commandes qui suivent, la fonction `par` permet d'ajuster des paramètres graphiques. En l'occurrence, ici, il s'agit de placer quatre histogrammes sur le même graphique, deux en ligne et deux en colonne.

```
hist(emp$logsalact, prob=T, main = "Logarithme du salaire")
curve(dnorm (x, mean=mean(emp$logsalact, na.rm = T),
             sd=sd(emp$logsalact, na.rm = T)), col=2, add = TRUE)
hist(emp$rcsalact, prob=T, main = "Racine carrée du salaire")
curve(dnorm (x, mean=mean(emp$rcsalact, na.rm = T),
             sd=sd(emp$rcsalact, na.rm = T)), col=2, add = TRUE)
```



Graphiquement, il semble que ce soit la transformation “inverse” qui permet au mieux de se rapprocher d’une distribution normale. Cette impression est confirmée par les valeurs des indicateurs d’asymétrie et d’aplatissement :

```
psych::describe (emp [, c("salact", "isalact", "logsalact", "rcsalact")])
```

```
##           vars   n     mean       sd   median  trimmed     mad      min
## salact       1 474 34419.57 17075.66 28875.00 31199.14 8562.01 15750.00
## isalact      2 474     0.00     0.00     0.00     0.00    0.00     0.00
## logsalact    3 474    10.36     0.40    10.27    10.31    0.30     9.66
## rcsalact     4 474   181.17    40.01   169.93   175.10   25.83   125.50
##                max     range  skew kurtosis     se
## salact    135000.00 119250.00  2.11     5.27 784.31
## isalact        0.00      0.00 -0.05    -0.27   0.00
## logsalact     11.81      2.15  0.99     0.65   0.02
## rcsalact     367.42    241.92  1.51     2.26   1.84
```

Chapter 8

Analyses bivariées

Les analyses bivariées permettent d'étudier les liens, les relations entre deux variables. Les analyses différent essentiellement en fonction du niveau de mesure des variables.

8.1 Analyses bivariées pour variables nominales

Le tableau de données "vote92.csv" (téléchargement) contient les résultats d'un sondage aux élections nord-américaines de 1992. La variable "pres92" contient l'expression des intentions en faveur des trois principaux candidats et la variable "agecat" la catégorie d'âge des répondants.

```
vote92 <- read.csv2 ("vote92.csv")
str (vote92)
```

```
## 'data.frame':    1847 obs. of  6 variables:
##  $ pres92: Factor w/ 3 levels "Bush","Clinton",..: 2 2 2 2 2 2 2 2 2 2 ...
##  $ age   : int  79 32 50 56 51 48 29 40 46 37 ...
##  $ agecat: Factor w/ 4 levels " lt 35","35 - 44",..: 4 1 3 3 3 3 1 2 3 2 ...
##  $ educ  : int  12 17 6 8 17 12 13 13 13 19 ...
##  $ degree: Factor w/ 5 levels "bachelor","graduate degree",..: 3 1 5 5 1 3 3 3 3 2 ...
##  $ sex   : Factor w/ 2 levels "female","male": 2 2 1 1 1 2 1 1 1 1 ...
```

8.1.1 Fonctions de base

Les fonctions `table` et `prop.table` fournissent respectivement les effectifs et les pourcentages dans un tableau croisé :

```
table (vote92$agecat, vote92$pres92)
```

```
##
##           Bush Clinton Perot
##    lt 35   153     186    99
##   35 - 44  156     207    81
##   45 - 64  219     316    82
##   65 +     133     199    16
```

```
prop.table (table (vote92$agecat, vote92$pres92), 1)  # proportions calculées par ligne
```

```
##
##                 Bush    Clinton      Perot
##    lt 35  0.34931507 0.42465753 0.22602740
##   35 - 44 0.35135135 0.46621622 0.18243243
##   45 - 64 0.35494327 0.51215559 0.13290113
##   65 +    0.38218391 0.57183908 0.04597701
```

```
prop.table (table (vote92$agecat, vote92$pres92), 1)  # proportions calculées par colonne
```

```
##
##                 Bush    Clinton      Perot
##    lt 35  0.34931507 0.42465753 0.22602740
##   35 - 44 0.35135135 0.46621622 0.18243243
##   45 - 64 0.35494327 0.51215559 0.13290113
##   65 +    0.38218391 0.57183908 0.04597701
```

Appliquée à la fonction `table`, la fonction générique `summary` calcule le χ^2, le nombre de degrés de libertés et la significativité.

```
summary (table (vote92$agecat, vote92$pres92))
```

```
## Number of cases in table: 1847
## Number of factors: 2
## Test for independence of all factors:
##  Chisq = 56.53, df = 6, p-value = 2.273e-10
```

8.1.2 Fonction `CrossTable` (paquet `gmodels`)

Ces fonctions de base ne permettent pas d'avoir les résultats présentés de manière très lisible. On pourra donc préférer la fonction `CrossTable` du paquet `gmodels` (Warnes, Bolker, Lumley, & Johnson (2015)) :

```
library (gmodels)
```

```
## Warning: le package 'gmodels' a été compilé avec la version R 3.4.3
```

```
CrossTable(vote92$agecat, vote92$pres92, chisq = TRUE)
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## | Chi-square contribution |
## |           N / Row Total |
## |           N / Col Total |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  1847
##
##
##                | vote92$pres92
```

```
## vote92$agecat |      Bush |   Clinton |     Perot | Row Total |
## --------------|-----------|-----------|-----------|-----------|
##         lt 35 |       153 |       186 |        99 |       438 |
##               |     0.090 |     3.994 |    16.594 |           |
##               |     0.349 |     0.425 |     0.226 |     0.237 |
##               |     0.231 |     0.205 |     0.356 |           |
##               |     0.083 |     0.101 |     0.054 |           |
## --------------|-----------|-----------|-----------|-----------|
##       35 - 44 |       156 |       207 |        81 |       444 |
##               |     0.053 |     0.582 |     3.005 |           |
##               |     0.351 |     0.466 |     0.182 |     0.240 |
##               |     0.236 |     0.228 |     0.291 |           |
##               |     0.084 |     0.112 |     0.044 |           |
## --------------|-----------|-----------|-----------|-----------|
##       45 - 64 |       219 |       316 |        82 |       617 |
##               |     0.015 |     0.530 |     1.272 |           |
##               |     0.355 |     0.512 |     0.133 |     0.334 |
##               |     0.331 |     0.348 |     0.295 |           |
##               |     0.119 |     0.171 |     0.044 |           |
## --------------|-----------|-----------|-----------|-----------|
##          65 + |       133 |       199 |        16 |       348 |
##               |     0.574 |     4.557 |    25.266 |           |
##               |     0.382 |     0.572 |     0.046 |     0.188 |
##               |     0.201 |     0.219 |     0.058 |           |
##               |     0.072 |     0.108 |     0.009 |           |
## --------------|-----------|-----------|-----------|-----------|
##  Column Total |       661 |       908 |       278 |      1847 |
##               |     0.358 |     0.492 |     0.151 |           |
## --------------|-----------|-----------|-----------|-----------|
##
##
## Statistics for All Table Factors
##
##
## Pearson's Chi-squared test
## ------------------------------------------------------------
## Chi^2 =  56.5313     d.f. =  6     p =  2.27288e-10
##
##
##
```

8.1.3 Coefficients d'associations

Les coefficients d'association dérivés du χ^2 ne sont fournis par aucune des fonctions précédentes. Ils peuvent se calculer facilement "à la main" en utilisant les formules suivantes :

$$phi = \sqrt{\frac{\chi^2}{n}}$$

```
sqrt (56.53/1847) # coefficient phi
```

```
## [1] 0.1749468
```

$$coefficient\ de\ contingence = \sqrt{\frac{\chi^2}{\chi^2 + n}}$$

```
sqrt (56.53/(1847 + 56.53)) # coefficient de contingence
```

```
## [1] 0.1723295
```

$$v\ de\ Cramér = \sqrt{\frac{\chi^2}{n \times min(c-1; l-1)}}$$

```
sqrt (56.53/(1847 * 2)) # coefficient v de Cramér
```

```
## [1] 0.1237061
```

Les moins courageux pourront apprécier la fonction `assocstats`[1] du paquet `vcd` (Zeileis, Meyer, & Hornik (2007)) :

```
library (vcd)
```

```
## Le chargement a nécessité le package : grid
```

```
assocstats (table (vote92$agecat, vote92$pres92))
```

```
##                     X^2 df   P(> X^2)
## Likelihood Ratio 63.654  6 8.1197e-12
## Pearson          56.531  6 2.2729e-10
##
## Phi-Coefficient   : NA
## Contingency Coeff.: 0.172
## Cramer's V        : 0.124
```

8.1.4 Ne pas confondre significativité et intensité

Le tableau `n390` contient des données fictives avec 390 observations et deux variables dichotomiques ("X1" et "X2").

```
n390 <- data.frame (cbind (c(rep ("oui", 200), rep ("non", 190)),
                           c(rep ("oui", 150), rep ("non", 50), rep ("oui", 100), rep ("non", 90))))
```

Le tableau croisé, le χ^2 et les coefficients d'association sont fournies par les commandes suivantes :

```
CrossTable (n390$X1, n390$X2,prop.r = F, prop.t = F)
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## | Chi-square contribution |
## |           N / Col Total |
## |-------------------------|
##
##
## Total Observations in Table:  390
```

[1] Mais cette fonction ne calcule le coefficient phi que pour les tableaux comprenant deux lignes et deux colonnes.

```
##
##
##              | n390$X2
##      n390$X1 |       non |       oui | Row Total |
## -------------|-----------|-----------|-----------|
##          non |        90 |       100 |       190 |
##              |     6.965 |     3.900 |           |
##              |     0.643 |     0.400 |           |
## -------------|-----------|-----------|-----------|
##          oui |        50 |       150 |       200 |
##              |     6.616 |     3.705 |           |
##              |     0.357 |     0.600 |           |
## -------------|-----------|-----------|-----------|
## Column Total |       140 |       250 |       390 |
##              |     0.359 |     0.641 |           |
## -------------|-----------|-----------|-----------|
##
##
```

```
assocstats (table (n390$X1, n390$X2))
```

```
##                     X^2 df   P(> X^2)
## Likelihood Ratio 21.401  1 3.7264e-06
## Pearson          21.186  1 4.1678e-06
##
## Phi-Coefficient   : 0.233
## Contingency Coeff.: 0.227
## Cramer's V        : 0.233
```

Le χ^2 est très significatif et l'association entre les deux variables n'est pas très élevée (0.23 pour le coefficient phi).

Si on prend maintenant le même tableau, avec 10 fois moins d'observations (39 au lieu de 390) :

```
n39 <- data.frame (cbind (c(rep ("oui", 20), rep ("non", 19)),
                          c(rep ("oui", 15), rep ("non", 5), rep ("oui", 10), rep ("non", 9))))
CrossTable (n39$X1, n39$X2,prop.r = F, prop.t = F)
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## | Chi-square contribution |
## |           N / Col Total |
## |-------------------------|
##
##
## Total Observations in Table:  39
##
##
##              | n39$X2
##       n39$X1 |       non |       oui | Row Total |
```

```
## -------------|-----------|-----------|-----------|
##          non |         9 |        10 |        19 |
##              |     0.696 |     0.390 |           |
##              |     0.643 |     0.400 |           |
## -------------|-----------|-----------|-----------|
##          oui |         5 |        15 |        20 |
##              |     0.662 |     0.371 |           |
##              |     0.357 |     0.600 |           |
## -------------|-----------|-----------|-----------|
## Column Total |        14 |        25 |        39 |
##              |     0.359 |     0.641 |           |
## -------------|-----------|-----------|-----------|
##
##
```

```
assocstats (table (n39$X1, n39$X2))
```

```
##                     X^2 df P(> X^2)
## Likelihood Ratio 2.1401  1  0.14350
## Pearson          2.1186  1  0.14552
##
## Phi-Coefficient   : 0.233
## Contingency Coeff.: 0.227
## Cramer's V        : 0.233
```

Les pourcentages dans les cellules du tableau sont strictement les mêmes, ainsi que la valeur des coefficients d'association : l'*intensité* (*effect size* en anglais) de la relation entre les deux variables est identique. Mais comme il y a beaucoup moins d'observations, le χ^2 n'est pas statistiquement significatif.

8.2 Analyses bivariées pour variables numériques

Le coefficient de corrélation linéaire (de Bravais-Pearson) est le coefficient traditionnel pour quantifier la relation entre deux variables numériques. Il s'obtient avec la fonction `cor` :

```
n250 <- read.csv2("notes250.csv")
n250 [,3:8] [n250 [,3:8] < 0] <- NA # Déclaration des données manquantes
cor (n250$cours1, n250$cours2, use = "complete.obs")
```

```
## [1] 0.5912525
```

Il est important de noter l'argument `use = "complete.obs"` dans la commande précédente. S'il n'est pas mentionné, étant donné que les variables comportent des données manquantes, le calcul n'est pas effectué :

```
cor (n250$cours1, n250$cours2)
```

```
## [1] NA
```

La significativité du coefficient de corrélation s'obtient avec la fonction `cor.test`[2] :

```
cor.test (n250$cours1, n250$cours2)
```

```
##
##  Pearson's product-moment correlation
##
```

[2]Par défaut, cette fonction exclut toutes les observations incomplètes.

```
## data:  n250$cours1 and n250$cours2
## t = 10.997, df = 225, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.4994957 0.6698909
## sample estimates:
##       cor
## 0.5912525
```

On peut calculer les corrélations entre plusieurs variables simultanément. Il y a alors deux options principales pour gérer les données manquantes :

- option "listwise" : les corrélations ne sont calculées que sur les observations complètes à toutes les variables ;
- option "pairwise" : les corrélations sont calculées sur les observations complètes pour chaque paire de variables (le nombre d'observations est donc potentiellement différent pour chaque corrélation) ;

```
cor(n250[,3:5], use="complete.obs") # option "listwise"
```

```
##           cours1    cours2    cours3
## cours1 1.0000000 0.5967601 0.2687736
## cours2 0.5967601 1.0000000 0.3804163
## cours3 0.2687736 0.3804163 1.0000000
```

```
cor(n250[,3:5], use="pairwise.complete.obs") # option "pairwise"
```

```
##           cours1    cours2    cours3
## cours1 1.0000000 0.5912525 0.2725388
## cours2 0.5912525 1.0000000 0.3862049
## cours3 0.2725388 0.3862049 1.0000000
```

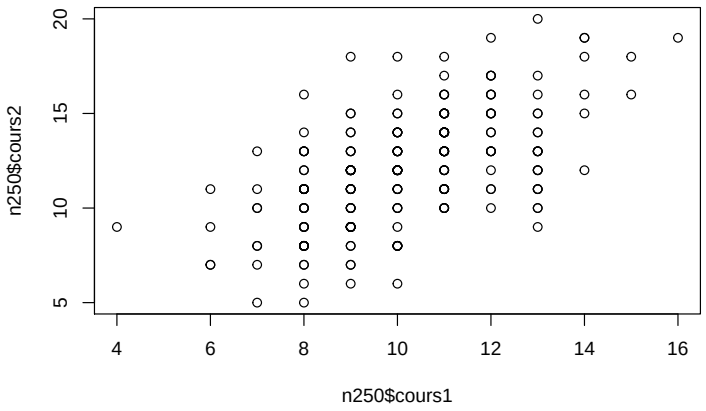
La fonction `cor.test` n'admet comme argument que deux variables : il n'est donc pas possible d'obtenir avec le module de base de `R` les significativités associées à chaque coefficient d'une matrice. On pourra donc préférer la fonction `corr.test` du package `psych` qui fournit les corrélations, les significativités et les effectifs.

```
library (psych)
corr.test (n250[,3:5], use = "pairwise")
```

```
## Call:corr.test(x = n250[, 3:5], use = "pairwise")
## Correlation matrix
##        cours1 cours2 cours3
## cours1   1.00   0.59   0.27
## cours2   0.59   1.00   0.39
## cours3   0.27   0.39   1.00
## Sample Size
##        cours1 cours2 cours3
## cours1    239    227    225
## cours2    227    237    222
## cours3    225    222    235
## Probability values (Entries above the diagonal are adjusted for multiple tests.)
##        cours1 cours2 cours3
## cours1      0      0      0
## cours2      0      0      0
## cours3      0      0      0
##
##  To see confidence intervals of the correlations, print with the short=FALSE option
```

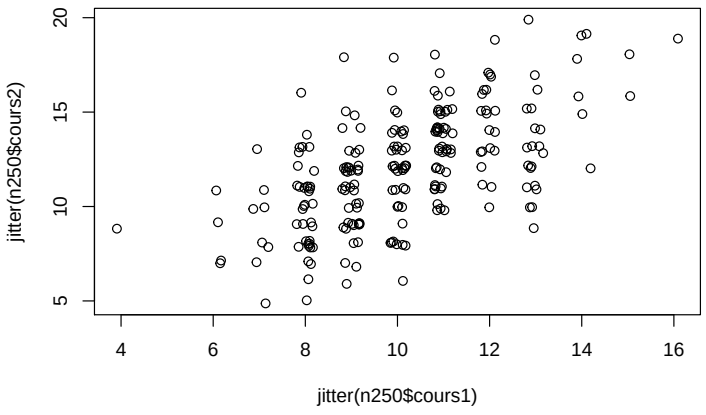
La représentation graphique du nuage de points entre deux variables est obtenue facilement avec la commande `plot` :

```
par (mfrow = c(1,1))
plot (n250$cours1, n250$cours2)
```



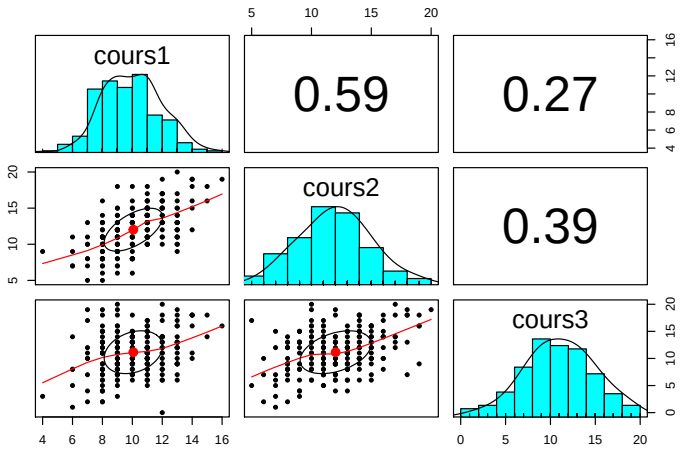
Dans ce nuage, un même point peut-être occupé par plusieurs observations (i.e. il y a plusieurs étudiants qui ont la note de 10 aux deux examens). L'utilisation de la fonction `jitter` qui décale la position des points peut alors être très utile :

```
plot (jitter (n250$cours1), jitter(n250$cours2))
```



Des représentations graphiques des nuages de points entre plusieurs variables, accompagnées de leur histogramme et des coefficients de corrélations correspondants peuvent être obtenues avec la fonction `pairs.panels` du paquet `psych` :

```
pairs.panels (n250[,3:5])
```



8.3 Analyses bivariées pour variables ordinales

Même si on assimile souvent les variables ordinales à des variables numériques (et ainsi, on calcule un coefficient de corrélation linéaire), il existe des coefficients dédiés aux variables ordinales, notamment le coefficient de corrélation par rangs ρ de Spearmann ou le coefficient τ de Kendall. Les fonctions à utiliser sont les mêmes que ci-dessus, mais il faut indiquer explicitement que l'on souhaite le coefficient de Spermann et/ou le coefficient de Kendall.

```
# le coefficient de Spearman
cor (n250$cours1, n250$cours2, use = "complete.obs", method = "spearman")
```

```
## [1] 0.5836488
```

```
cor.test(n250$cours1, n250$cours2, method = "spearman")
```

```
## Warning in cor.test.default(n250$cours1, n250$cours2, method = "spearman"):
## Cannot compute exact p-value with ties
```

```
##
##  Spearman's rank correlation rho
##
## data:  n250$cours1 and n250$cours2
## S = 811670, p-value < 2.2e-16
## alternative hypothesis: true rho is not equal to 0
## sample estimates:
##       rho
## 0.5836488
```

```
cor(n250[,3:5], use="pairwise.complete.obs", method = "spearman")
```

```
##           cours1    cours2    cours3
## cours1 1.0000000 0.5836488 0.2182171
## cours2 0.5836488 1.0000000 0.3378869
## cours3 0.2182171 0.3378869 1.0000000
```

```r
# Avec le paquet `psych`
corr.test (n250[,3:5], use = "pairwise", method = "spearman")
```

```
## Call:corr.test(x = n250[, 3:5], use = "pairwise", method = "spearman")
## Correlation matrix
##        cours1 cours2 cours3
## cours1   1.00   0.58   0.22
## cours2   0.58   1.00   0.34
## cours3   0.22   0.34   1.00
## Sample Size
##        cours1 cours2 cours3
## cours1    239    227    225
## cours2    227    237    222
## cours3    225    222    235
## Probability values (Entries above the diagonal are adjusted for multiple tests.)
##        cours1 cours2 cours3
## cours1      0      0      0
## cours2      0      0      0
## cours3      0      0      0
##
##  To see confidence intervals of the correlations, print with the short=FALSE option
```

```r
# le coefficient de Kendall
cor (n250$cours1, n250$cours2, use = "complete.obs", method = "kendall")
```

```
## [1] 0.4587585
```

```r
cor.test(n250$cours1, n250$cours2, method = "kendall")
```

```
##
##  Kendall's rank correlation tau
##
## data:  n250$cours1 and n250$cours2
## z = 9.2268, p-value < 2.2e-16
## alternative hypothesis: true tau is not equal to 0
## sample estimates:
##       tau
## 0.4587585
```

```r
cor(n250[,3:5], use="pairwise.complete.obs", method = "kendall")
```

```
##           cours1    cours2    cours3
## cours1 1.0000000 0.4587585 0.1667994
## cours2 0.4587585 1.0000000 0.2551679
## cours3 0.1667994 0.2551679 1.0000000
```

```r
# Avec le paquet `psych`
corr.test (n250[,3:5], use = "pairwise", method = "kendall")
```

```
## Call:corr.test(x = n250[, 3:5], use = "pairwise", method = "kendall")
```

```
## Correlation matrix
##        cours1 cours2 cours3
## cours1   1.00   0.46   0.17
## cours2   0.46   1.00   0.26
## cours3   0.17   0.26   1.00
## Sample Size
##        cours1 cours2 cours3
## cours1    239    227    225
## cours2    227    237    222
## cours3    225    222    235
## Probability values (Entries above the diagonal are adjusted for multiple tests.)
##        cours1 cours2 cours3
## cours1   0.00      0   0.01
## cours2   0.00      0   0.00
## cours3   0.01      0   0.00
##
##  To see confidence intervals of the correlations, print with the short=FALSE option
```

8.4 Analyses bivariées pour une variable dépendante numérique et une variable indépendante dichotomique (test t de Student)

Un cas particulier des analyses bivariées se présente lorsqu'on a une variable dépendante numérique et une variable indépendante dichotomique. On utilise alors le test `t` de Student. Lorsque la variable indépendante comporte plus de deux modalités, on a recours à l'analyse de variance à laquelle sera consacré un chapitre entier ultérieurement.

L'utilisation de la fonction `t.test` est assez simple :

```
t.test (n250$cours1~n250$sexe)
```

```
##
##  Welch Two Sample t-test
##
## data:  n250$cours1 by n250$sexe
## t = 0.55659, df = 69.767, p-value = 0.5796
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -0.435709  0.773005
## sample estimates:
## mean in group 1 mean in group 2
##        10.20455        10.03590
```

Par défaut, cette fonction R fournit les résultats du test de Welch qui est une adaptation du test de Student lorsque les variances sont inégales dans les deux groupes. Pour obtenir un "vrai" test de Student, il faut utiliser l'argument `var.equal` :

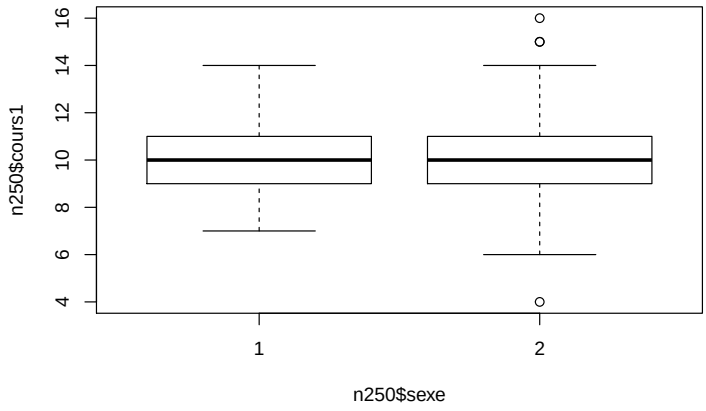
```
t.test (n250$cours1~n250$sexe, var.equal = T)
```

```
##
##  Two Sample t-test
##
## data:  n250$cours1 by n250$sexe
## t = 0.51684, df = 237, p-value = 0.6057
## alternative hypothesis: true difference in means is not equal to 0
```

```
## 95 percent confidence interval:
##  -0.4741798  0.8114758
## sample estimates:
## mean in group 1 mean in group 2
##        10.20455        10.03590
```

La fonction `plot` fournit un graphique de base : boîtes de dispersion ou "*box-plot*" (médiane[3], intervalle inter-quartiles, intervalle de confiance autour de la médiane et points extrêmes) :

```
n250$sexe <- factor (n250$sexe) # La variable indépendante doit être définie comme un "facteur"
plot(n250$cours1 ~ n250$sexe)
```



[3]Attention, il s'agit bien des médianes qui sont produites par cette commande et pas les moyennes.

Chapter 9

Analyses en composantes principales et en facteurs communs

La présentation de l'analyse en composantes principales et l'analyse en facteurs communs s'appuie sur les résultats à 8 tests d'aptitudes administrés à un échantillon de 181 adolescents :

- LECT : vitesse de lecture
- CARPLI : carrés pliés (on présente le dessin d'un carré de papier plié en 4 que l'on a entaillé avec des ciseaux : on présente le dessin d'un carré déplié ; le sujet doit dessiner ce que l'on obtiendrait si on dépliait le carré plié)
- B53 : séries à compléter (du type matrices de Raven)
- VOC : test de vocabulaire (trouver un synonyme)
- PFB : Paper Form Board (on présente des morceaux de figure et le sujet doit choisir la figure qui serait construite si on assemblait les morceaux)
- NUM : séries de nombres à compléter
- ANAL : analogies verbales
- D70 : séries de dominos à compléter

On ne dispose pas des données brutes pour cette étude, mais uniquement des corrélations entre les tests :

	lect	carpli	b53	voc	pfb	num	anal	d70
lect	1.000	0.252	0.408	0.571	0.307	0.356	0.506	0.271
carpli	0.252	1.000	0.623	0.290	0.437	0.365	0.281	0.411
b53	0.408	0.623	1.000	0.415	0.428	0.478	0.473	0.512
voc	0.571	0.290	0.415	1.000	0.313	0.394	0.610	0.336
pfb	0.307	0.437	0.428	0.313	1.000	0.289	0.295	0.376
num	0.356	0.365	0.478	0.394	0.289	1.000	0.495	0.441
anal	0.506	0.281	0.473	0.610	0.295	0.495	1.000	0.441
d70	0.271	0.411	0.512	0.336	0.376	0.441	0.441	1.000

Cette matrice de corrélations peut être définie comme un objet `matrice` dans `R` avec les commandes suivantes.[1] L'analyse en composantes principales ou en facteurs communs peut ensuite porter directement sur cette matrice.

[1] Pour éviter la saisie fastidieuse des corrélations, on peut importer le fichier "matcor.csv" (téléchargement) sous le nom de "cortest" (`cortest <- read.csv2("matcor.csv")`), puis le transformer en matrice (`cortest <- as.matrix (cortest)`) et enfin donner les noms de variables aux lignes de la matrice (`rownames (cortest) <- colnames (cortest)`)

```
cortest <- matrix (c(1, .252, .408, .571, .307, .356, .506, .271,
            .252, 1, .623, .290, .437, .365, .281, .411,
            .408, .623, 1, .415, .428, .478, .473, .512,
            .571, .290,  .415, 1, .313, .394, .610, .336,
            .307, .437, .428, .313, 1, .289, .295, .376,
            .356, .365, .478, .394, .289, 1, .495, .441,
            .506, .281, .473, .610, .295, .495, 1, .441,
            .271, .411, .512, .336, .376, .441, .441, 1),
nrow = 8, ncol = 8, byrow = T)
# donner des noms aux lignes de la matrice
rownames (cortest) <- c ("lect", "carpli", "b53", "voc", "pfb", "num", "anal", "d70")
# donner les mêmes noms aux colonnes
colnames (cortest) <- rownames (cortest)
cortest
```

```
##         lect carpli   b53   voc   pfb   num  anal   d70
## lect   1.000  0.252 0.408 0.571 0.307 0.356 0.506 0.271
## carpli 0.252  1.000 0.623 0.290 0.437 0.365 0.281 0.411
## b53    0.408  0.623 1.000 0.415 0.428 0.478 0.473 0.512
## voc    0.571  0.290 0.415 1.000 0.313 0.394 0.610 0.336
## pfb    0.307  0.437 0.428 0.313 1.000 0.289 0.295 0.376
## num    0.356  0.365 0.478 0.394 0.289 1.000 0.495 0.441
## anal   0.506  0.281 0.473 0.610 0.295 0.495 1.000 0.441
## d70    0.271  0.411 0.512 0.336 0.376 0.441 0.441 1.000
```

9.1 Analyse en composantes principales

Le package `psych` (Revelle (2017)) contient la fonction `principal` qui produit les principaux résultats d'une analyse en composantes principales (ACP) telle que la pratiquent habituellement les psychologues, avec ou sans rotation. Les résultats de l'analyse sont stockés dans l'objet `acpcortest`[2].

```
library (psych)
acpcortest <- principal (cortest, nfactors = 2, residuals = T, rotate = "varimax")
```

Avant de décrire le contenu de l'objet `acpcortest`, voyons quelques uns des arguments de la fonction `principal` :

- L'argument `nfactors = 2` indique le nombre de facteurs souhaités ;
- L'argument `rotate = "none"` fournit les résultats avant rotation (autres rotations possibles : varimax, quatimax, promax, oblimin, simplimax, cluster) ;
- l'argument `residuals = F` supprime le calcul des résidus ;

L'objet `acpcortest` est un objet de type liste, comprenant pas moins de 28 éléments (dont certains sont eux-mêmes des listes décomposées en plusieurs sous-éléments) :

```
str (acpcortest)
```

```
## List of 28
##  $ values      : num [1:8] 3.864 1.105 0.75 0.606 0.531 ...
##  $ rotation    : chr "varimax"
##  $ n.obs       : logi NA
```

[2]Tous les résultats d'analyses (univariées, bivariées, multivariées) peuvent être stockés dans un objet. Jusqu'à présent, on n'a pas utilisé souvent cette possibilité, parce que les résultats se limitaient souvent à quelques lignes ou à un ou deux tableaux. Avec les analyses multivariées, les résultats sont beaucoup plus volumineux et il est plus simple de les stocker dans un objet spécifique d'où on peut extraire les informations que l'on souhaite une ou à une

```
##  $ communality : Named num [1:8] 0.649 0.706 0.701 0.722 0.479 ...
##   ..- attr(*, "names")= chr [1:8] "lect" "carpli" "b53" "voc" ...
##  $ loadings    : loadings [1:8, 1:2] 0.157 0.837 0.762 0.195 0.671 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:8] "lect" "carpli" "b53" "voc" ...
##   .. ..$ : chr [1:2] "RC1" "RC2"
##  $ residual    : num [1:8, 1:8] 0.3513 0.0647 0.0137 -0.1131 0.0663 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:8] "lect" "carpli" "b53" "voc" ...
##   .. ..$ : chr [1:8] "lect" "carpli" "b53" "voc" ...
##  $ fit         : num 0.907
##  $ fit.off     : num 0.956
##  $ fn          : chr "principal"
##  $ Call        : language principal(r = cortest, nfactors = 2, residuals = T, rotate = "varimax")
##  $ uniquenesses: Named num [1:8] 0.351 0.294 0.299 0.278 0.521 ...
##   ..- attr(*, "names")= chr [1:8] "lect" "carpli" "b53" "voc" ...
##  $ complexity  : Named num [1:8] 1.08 1.01 1.4 1.11 1.13 ...
##   ..- attr(*, "names")= chr [1:8] "lect" "carpli" "b53" "voc" ...
##  $ chi         : num NA
##  $ EPVAL       : num NA
##  $ R2          : Named num [1:2] 1 1
##   ..- attr(*, "names")= chr [1:2] "RC1" "RC2"
##  $ objective   : num 0.414
##  $ rms         : num 0.0873
##  $ factors     : int 2
##  $ dof         : num 13
##  $ null.dof    : num 28
##  $ null.model  : num 2.89
##  $ criteria    : Named num [1:3] 0.414 NA NA
##   ..- attr(*, "names")= chr [1:3] "objective" "" ""
##  $ PVAL        : logi NA
##  $ weights     : num [1:8, 1:2] -0.166 0.453 0.32 -0.156 0.326 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:8] "lect" "carpli" "b53" "voc" ...
##   .. ..$ : chr [1:2] "RC1" "RC2"
##  $ r.scores    : num [1:2, 1:2] 1.00 -5.27e-16 -5.62e-16 1.00
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:2] "RC1" "RC2"
##   .. ..$ : chr [1:2] "RC1" "RC2"
##  $ rot.mat     : num [1:2, 1:2] 0.722 0.692 -0.692 0.722
##  $ Vaccounted  : num [1:5, 1:2] 2.545 0.318 0.318 0.512 0.512 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:5] "SS loadings" "Proportion Var" "Cumulative Var" "Proportion Explained" ...
##   .. ..$ : chr [1:2] "RC1" "RC2"
##  $ Structure   : loadings [1:8, 1:2] 0.157 0.837 0.762 0.195 0.671 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:8] "lect" "carpli" "b53" "voc" ...
##   .. ..$ : chr [1:2] "RC1" "RC2"
##  - attr(*, "class")= chr [1:2] "psych" "principal"
```

Il serait fastidieux de les décrire en détail. Les éléments les plus importants sont :

- `values` (les valeurs-propres) ;
- `loadings` (les saturations) ;
- `communality` (les communautés) ;

- `residual` (les résidus) ;
- `scores` (les scores factoriels si ceux-ci sont demandés et uniquement si l'analyse a porté sur les données brutes) ;
- `phi` (les corrélations entre les facteurs, si une rotation oblique est demandée) ;

L'affichage d'un élément se fait un indiquant le nom de l'objet suivi du signe `$` et ensuite du nom de l'élément. Par exemple, pour obtenir les saturations factorielles, on écrira :

```
acpcortest$loadings
```

```
##
## Loadings:
##        RC1   RC2
## lect   0.157 0.790
## carpli 0.837
## b53    0.762 0.348
## voc    0.195 0.827
## pfb    0.671 0.171
## num    0.481 0.496
## anal   0.278 0.793
## d70    0.665 0.292
##
##                  RC1   RC2
## SS loadings    2.545 2.424
## Proportion Var 0.318 0.303
## Cumulative Var 0.318 0.621
```

On peut remarquer que certaines saturations n'apparaissent pas : il s'agit des saturations proches de 0 (inférieures à 0.10 en valeur absolue). Pour visualiser toutes les saturations, on peut appliquer la fonction `unclass` :

```
unclass (acpcortest$loadings)
```

```
##              RC1        RC2
## lect   0.1571154 0.78997313
## carpli 0.8373506 0.07059923
## b53    0.7616905 0.34766142
## voc    0.1951243 0.82711778
## pfb    0.6708339 0.17122832
## num    0.4811176 0.49636998
## anal   0.2783458 0.79330950
## d70    0.6646715 0.29215310
```

Les valeurs-propres (arrondies) :

```
round (acpcortest$value,1)
```

```
## [1] 3.9 1.1 0.7 0.6 0.5 0.5 0.4 0.3
```

Les résidus (arrondis) :

```
round (acpcortest$residual,2)
```

```
##         lect carpli   b53   voc   pfb   num  anal   d70
## lect    0.35   0.06  0.01 -0.11  0.07 -0.11 -0.16 -0.06
## carpli  0.06   0.29 -0.04  0.07 -0.14 -0.07 -0.01 -0.17
## b53     0.01  -0.04  0.30 -0.02 -0.14 -0.06 -0.01 -0.10
## voc    -0.11   0.07 -0.02  0.28  0.04 -0.11 -0.10 -0.04
## pfb     0.07  -0.14 -0.14  0.04  0.52 -0.12 -0.03 -0.12
```

```
## num    -0.11  -0.07 -0.06 -0.11 -0.12  0.52 -0.03 -0.02
## anal   -0.16  -0.01 -0.01 -0.10 -0.03 -0.03  0.29  0.02
## d70    -0.06  -0.17 -0.10 -0.04 -0.12 -0.02  0.02  0.47
```

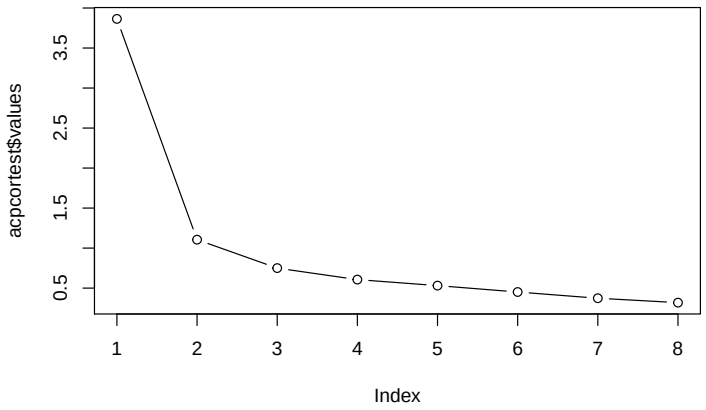
Les communautés (arrondies) :

```
round (acpcortest$communality,2)
```

```
##   lect carpli    b53    voc    pfb    num   anal    d70
##   0.65   0.71   0.70   0.72   0.48   0.48   0.71   0.53
```

Le graphique des valeurs propres est disponible grâce à la fonction générique `plot` qui comprend de nombreux arguments potentiels. L'un d'entre eux est utilisé ici (`type = "b"`), ce qui permet d'obtenir un graphique comportant à la fois des points et des lignes) :

```
plot (acpcortest$values, type = "b")
```



Les corrélations reproduites à partir de la solution factorielle ne sont pas disponibles dans l'objet `acpcortest`. Mais il est facile des les obtenir en multipliant la matrice des saturations par sa transposée[3]. La fonction `t` permet de transposer une matrice et la fonction `%*%` de multiplier deux matrices :

```
round (acpcortest$loadings %*% t (acpcortest$loadings),2)
```

```
##        lect carpli  b53  voc  pfb  num anal  d70
## lect   0.65   0.19 0.39 0.68 0.24 0.47 0.67 0.34
## carpli 0.19   0.71 0.66 0.22 0.57 0.44 0.29 0.58
## b53    0.39   0.66 0.70 0.44 0.57 0.54 0.49 0.61
## voc    0.68   0.22 0.44 0.72 0.27 0.50 0.71 0.37
## pfb    0.24   0.57 0.57 0.27 0.48 0.41 0.32 0.50
## num    0.47   0.44 0.54 0.50 0.41 0.48 0.53 0.46
## anal   0.67   0.29 0.49 0.71 0.32 0.53 0.71 0.42
## d70    0.34   0.58 0.61 0.37 0.50 0.46 0.42 0.53
```

En faisant la différence entre les corrélations observées et les corrélations reproduites, on retrouve bien les corrélations résiduelles fournies directement par la fonction `principal` :

```
# Différence entre les corrélations observées et les corrélations reproduites
round (cortest - acpcortest$loadings %*% t (acpcortest$loadings), 3)
```

[3] C'est-à-dire en inversant ses lignes et ses colonnes. Multiplier une matrice de saturations par sa transposée revient à faire "la somme du produit des saturations".

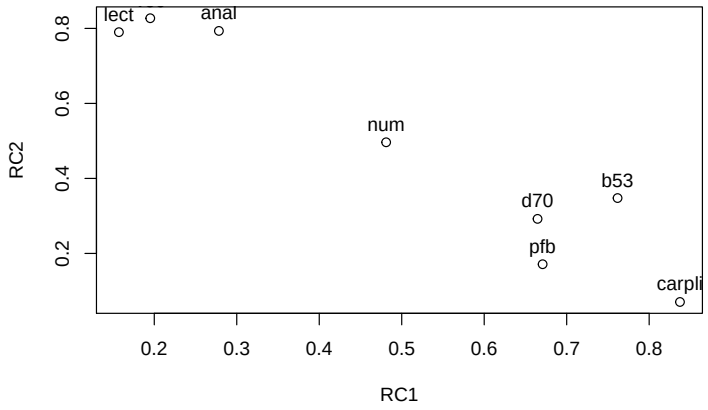
```
##          lect carpli    b53    voc    pfb    num   anal    d70
## lect    0.351  0.065  0.014 -0.113  0.066 -0.112 -0.164 -0.064
## carpli  0.065  0.294 -0.039  0.068 -0.137 -0.073 -0.008 -0.166
## b53     0.014 -0.039  0.299 -0.021 -0.142 -0.061 -0.015 -0.096
## voc    -0.113  0.068 -0.021  0.278  0.040 -0.110 -0.100 -0.035
## pfb     0.066 -0.137 -0.142  0.040  0.521 -0.119 -0.028 -0.120
## num    -0.112 -0.073 -0.061 -0.110 -0.119  0.522 -0.033 -0.024
## anal   -0.164 -0.008 -0.015 -0.100 -0.028 -0.033  0.293  0.024
## d70    -0.064 -0.166 -0.096 -0.035 -0.120 -0.024  0.024  0.473
```

```
# matrice des résidus
round (acpcortest$residual,3)
```

```
##          lect carpli    b53    voc    pfb    num   anal    d70
## lect    0.351  0.065  0.014 -0.113  0.066 -0.112 -0.164 -0.064
## carpli  0.065  0.294 -0.039  0.068 -0.137 -0.073 -0.008 -0.166
## b53     0.014 -0.039  0.299 -0.021 -0.142 -0.061 -0.015 -0.096
## voc    -0.113  0.068 -0.021  0.278  0.040 -0.110 -0.100 -0.035
## pfb     0.066 -0.137 -0.142  0.040  0.521 -0.119 -0.028 -0.120
## num    -0.112 -0.073 -0.061 -0.110 -0.119  0.522 -0.033 -0.024
## anal   -0.164 -0.008 -0.015 -0.100 -0.028 -0.033  0.293  0.024
## d70    -0.064 -0.166 -0.096 -0.035 -0.120 -0.024  0.024  0.473
```

La représentation graphique des deux premiers axes de la solution est à nouveau le résultat de la fonction `plot`. Pour ajouter les noms des variables, on utilise la fonction `text` avec deux arguments : `labels` pour indiquer le nom des variables et `pos` pour indiquer à quel endroit positionner les noms (`pos = 3` signifie que les noms seront positionnés au dessus des points).

```
plot (acpcortest$loadings)
text (acpcortest$loadings, labels = rownames (cortest), pos = 3)
```



Les mêmes commandes peuvent être utilisées pour obtenir la solution avant rotation (argument `rotate = none`)...

```
principal (cortest, nfactors = 2, residuals = T, rotate = "none")
```

```
## Principal Components Analysis
## Call: principal(r = cortest, nfactors = 2, residuals = T, rotate = "none")
```

```
## Standardized loadings (pattern matrix) based upon correlation matrix
##         PC1   PC2   h2   u2 com
## lect   0.66 -0.46 0.65 0.35 1.8
## carpli 0.65  0.53 0.71 0.29 1.9
## b53    0.79  0.28 0.70 0.30 1.2
## voc    0.71 -0.46 0.72 0.28 1.7
## pfb    0.60  0.34 0.48 0.52 1.6
## num    0.69 -0.03 0.48 0.52 1.0
## anal   0.75 -0.38 0.71 0.29 1.5
## d70    0.68  0.25 0.53 0.47 1.3
##
##                        PC1  PC2
## SS loadings           3.86 1.11
## Proportion Var        0.48 0.14
## Cumulative Var        0.48 0.62
## Proportion Explained  0.78 0.22
## Cumulative Proportion 0.78 1.00
##
## Mean item complexity =  1.5
## Test of the hypothesis that 2 components are sufficient.
##
## The root mean square of the residuals (RMSR) is  0.09
##
## Fit based upon off diagonal values = 0.96
```

... ou la solution factorielle avec une autre méthode de rotation (ici, par exemple une rotation oblique) :

```
principal (cortest, nfactors = 2, residuals = T, rotate = "promax")
```

```
## Principal Components Analysis
## Call: principal(r = cortest, nfactors = 2, residuals = T, rotate = "promax")
## Standardized loadings (pattern matrix) based upon correlation matrix
##          RC1   RC2   h2   u2 com
## lect   -0.11  0.86 0.65 0.35 1.0
## carpli  0.95 -0.22 0.71 0.29 1.1
## b53     0.76  0.13 0.70 0.30 1.1
## voc    -0.08  0.89 0.72 0.28 1.0
## pfb     0.72 -0.04 0.48 0.52 1.0
## num     0.38  0.40 0.48 0.52 2.0
## anal    0.03  0.82 0.71 0.29 1.0
## d70     0.66  0.10 0.53 0.47 1.0
##
##                        RC1  RC2
## SS loadings           2.55 2.42
## Proportion Var        0.32 0.30
## Cumulative Var        0.32 0.62
## Proportion Explained  0.51 0.49
## Cumulative Proportion 0.51 1.00
##
##  With component correlations of
##      RC1  RC2
## RC1 1.00 0.57
## RC2 0.57 1.00
##
## Mean item complexity =  1.2
```

```
## Test of the hypothesis that 2 components are sufficient.
##
## The root mean square of the residuals (RMSR) is  0.09
##
## Fit based upon off diagonal values = 0.96
```

9.2 Analyses en facteurs communs

Les analyses en facteurs communs sont disponibles (toujours dans le paquet `psych`) avec la fonction `fa`. Dans le cas où l'analyse porte directement sur une matrice de corrélations, les commandes sont les suivantes :

```
afccortest <- fa (cortest, fm = "ml", nfactors = 2, n.obs = 181, rotate = "varimax")
```

Quelques précisions quant aux arguments utilisés dans cette commande :

- `fm` : permet de préciser le type d'algorithme utilisé pour la solution factorielle. Plusieurs méthodes sont disponibles (parmi lesquelles) :
  - "ml" : maximum de vraisemblance
  - "pa" : axes principaux
  - "wls" : moindres carrés pondérés
  - "gls" : moindres carrés généralisés
  - "minres" : résidu minimum
- `n.obs` : cet argument (nombre d'observations) est lui aussi indispensable lorsqu'on analyse une matrice de corrélations. Lorsqu'on analyse les observations brutes, le logiciel détermine lui-même automatiquement cet élément ;
- `rotate` indique le type de rotation à appliquer à la solution factorielle. La solution avant rotation s'obtient en indiquant : `rotate = "none"`. De multiples méthodes sont disponibles : "varimax", "quartimax", "bentlerT", "equamax", "varimin", "geominT" et "bifactor" pour des rotations orthogonales ; "Promax", "promax", "oblimin", "simplimax", "bentlerQ,"geominQ","biquartimin" et "cluster" pour des rotations obliques.

En indiquant uniquement le nom de l'objet-résultat, on obtient une extraction des principaux résultats :

```
afccortest
```

```
## Factor Analysis using method =  ml
## Call: fa(r = cortest, nfactors = 2, n.obs = 181, rotate = "varimax",
##     fm = "ml")
## Standardized loadings (pattern matrix) based upon correlation matrix
##         ML1  ML2   h2   u2 com
## lect   0.23 0.64 0.46 0.54 1.2
## carpli 0.77 0.12 0.60 0.40 1.0
## b53    0.75 0.35 0.68 0.32 1.4
## voc    0.23 0.74 0.61 0.39 1.2
## pfb    0.50 0.24 0.31 0.69 1.5
## num    0.43 0.45 0.39 0.61 2.0
## anal   0.28 0.74 0.63 0.37 1.3
## d70    0.53 0.34 0.39 0.61 1.7
##
##                        ML1  ML2
## SS loadings           2.04 2.01
## Proportion Var        0.25 0.25
## Cumulative Var        0.25 0.51
## Proportion Explained  0.50 0.50
```

```
## Cumulative Proportion 0.50 1.00
##
## Mean item complexity =  1.4
## Test of the hypothesis that 2 factors are sufficient.
##
## The degrees of freedom for the null model are  28  and the objective function was  2.89 with Chi Squa
## The degrees of freedom for the model are 13  and the objective function was  0.1
##
## The root mean square of the residuals (RMSR) is  0.03
## The df corrected root mean square of the residuals is  0.05
##
## The harmonic number of observations is  181 with the empirical chi square  9.94  with prob <  0.7
## The total number of observations was  181  with Likelihood Chi Square =  16.77  with prob <  0.21
##
## Tucker Lewis Index of factoring reliability =  0.983
## RMSEA index =  0.043  and the 90 % confidence intervals are  0 0.089
## BIC =  -50.81
## Fit based upon off diagonal values = 0.99
## Measures of factor score adequacy
##                                                 ML1  ML2
## Correlation of scores with factors             0.87 0.87
## Multiple R square of scores with factors       0.76 0.75
## Minimum correlation of possible factor scores  0.51 0.50
```

On peut aussi, à l'instar de ce que l'on a vu avec l'analyse en composantes principales demander des résultats spécifiques venant compléter ce résumé. L'objet-résultat est une liste avec 47 éléments[4]. Parmi les plus importants, on peut citer[5] :

- `e.values` : les valeurs propres
- `communality` : les communautés
- `loadings` : la matrice factorielle
- `Phi` : les corrélations entre facteurs (en cas de rotation oblique)
- `residual` : les corrélations résiduelles[6]
- `scores` : les scores factoriels (si ceux-ci ont été demandés)

9.3 Analyses sur le tableau de données et scores factoriels

Dans les paragraphes précédents, nous avons vu les principales commandes pour réaliser des ACP ou des AFC à partir d'une matrice de corrélations. Il est plus fréquent de procéder à l'analyse directement sur le tableau de données des observations. Les données qui vont servir à illustrer cette possibilité sont des données classiquement utilisées pour présenter l'analyse factorielle (Holzinger & Swineford (1937)). Il s'agit de 9 tests d'aptitudes administrés à 301 enfants. Ces 9 tests sont :

- visp : perception visuelle
- cub : cubes
- los : losanges
- comprendre : compréhension de paragraphes
- completer : complétion de phrases
- voc : vocabulaire (signification)
- addition : additions simples

[4]Rappelons que la commande `str (afccortest)` permet d'accéder à la liste de tous les éléments

[5]On se rappelle que, pour y accéder, il suffit d'indiquer le nom de l'objet-résultat, le signe `$` et le nom de l'élément. Par exemple `afccortest$e.values`.

[6]On peut aussi les afficher sous forme de matrice triangulaire avec la commande : `residuals (afccortest)`.

- compterp : comptabilisation du nombre de points sur une page
- discrim : discriminer rapidement des majuscules avec des courbes (P) de majuscules avec des droites (E)

Le tableau de données s'appelle "holzinger.csv" (téléchargement)

```
holz <- read.csv2("holzinger.csv")
str (holz)
```

```
## 'data.frame':    301 obs. of  15 variables:
##  $ X         : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ id        : int  1 2 3 4 5 6 7 8 9 11 ...
##  $ school    : Factor w/ 2 levels "Grant-White",..: 2 2 2 2 2 2 2 2 2 2 ...
##  $ grade     : int  7 7 7 7 7 7 7 7 7 7 ...
##  $ visp      : num  3.33 5.33 4.5 5.33 4.83 ...
##  $ cub       : num  7.75 5.25 5.25 7.75 4.75 5 6 6.25 5.75 5.25 ...
##  $ los       : num  0.375 2.125 1.875 3 0.875 ...
##  $ comprendre: num  2.33 1.67 1 2.67 2.67 ...
##  $ completer : num  5.75 3 1.75 4.5 4 3 6 4.25 5.75 5 ...
##  $ voc       : num  1.286 1.286 0.429 2.429 2.571 ...
##  $ addition  : num  3.39 3.78 3.26 3 3.7 ...
##  $ compterp  : num  5.75 6.25 3.9 5.3 6.3 6.65 6.2 5.15 4.65 4.55 ...
##  $ discrim   : num  6.36 7.92 4.42 4.86 5.92 ...
##  $ agemois   : int  157 163 157 158 146 169 145 146 156 149 ...
##  $ sexe      : Factor w/ 2 levels "f","m": 2 1 1 2 1 1 2 1 1 1 ...
```

Traditionnellement, ces données sont analysées avec une structure en trois facteurs : aptitudes visuelles, aptitudes verbales et vitesse de traitement de l'information. Pour réaliser une ACP ou une AFC (ici en trois facteurs avec une rotation oblique), il suffit d'indiquer la partie du tableau de données contenant les noms des variables sur lesquelles doit porter l'analyse[7] :

```
holzacp <- principal (holz [,5:13], nfactors = 3, rotate = "promax")
holzafc <- fa (holz [,5:13], nfactors = 3, fm = "ml", rotate = "promax")
```

```
## Le chargement a nécessité le package : GPArotation
```

Les principaux résultats des deux analyses donnent des résultats assez semblables :

```
# résultats de l'ACP
holzacp
```

```
## Principal Components Analysis
## Call: principal(r = holz[, 5:13], nfactors = 3, rotate = "promax")
## Standardized loadings (pattern matrix) based upon correlation matrix
##              RC1   RC3   RC2   h2   u2 com
## visp        0.21  0.66  0.05 0.59 0.41 1.2
## cub        -0.02  0.77 -0.23 0.55 0.45 1.2
## los        -0.12  0.81  0.04 0.63 0.37 1.1
## comprendre  0.90 -0.01  0.00 0.81 0.19 1.0
## completer   0.93 -0.07  0.00 0.82 0.18 1.0
## voc         0.88  0.06 -0.02 0.79 0.21 1.0
## addition    0.05 -0.26  0.88 0.72 0.28 1.2
## compterp   -0.06  0.07  0.82 0.69 0.31 1.0
## discrim     0.00  0.39  0.58 0.61 0.39 1.7
##
##                        RC1  RC3  RC2
```

[7] En cas de données manquantes, les corrélations sont calculées avec l'option "pairwise". Les fonctions `principal` et `fa` comportent aussi des options pour imputer les données manquantes par la moyenne ou la médiane (voir l'argument `impute`).

```
## SS loadings           2.50 1.90 1.82
## Proportion Var        0.28 0.21 0.20
## Cumulative Var        0.28 0.49 0.69
## Proportion Explained  0.40 0.31 0.29
## Cumulative Proportion 0.40 0.71 1.00
##
##  With component correlations of
##      RC1  RC3  RC2
## RC1 1.00 0.32 0.22
## RC3 0.32 1.00 0.27
## RC2 0.22 0.27 1.00
##
## Mean item complexity =  1.2
## Test of the hypothesis that 3 components are sufficient.
##
## The root mean square of the residuals (RMSR) is  0.08
##  with the empirical chi square  149.47  with prob <  7.3e-26
##
## Fit based upon off diagonal values = 0.93
```

```
# résultats de l'AFC
holzafc
```

```
## Factor Analysis using method =  ml
## Call: fa(r = holz[, 5:13], nfactors = 3, rotate = "promax", fm = "ml")
## Standardized loadings (pattern matrix) based upon correlation matrix
##              ML1   ML2   ML3   h2   u2 com
## visp        0.15  0.04  0.61 0.49 0.51 1.1
## cub         0.01 -0.12  0.52 0.25 0.75 1.1
## los        -0.11  0.03  0.70 0.46 0.54 1.1
## comprendre  0.84  0.00  0.01 0.72 0.28 1.0
## completer   0.90  0.01 -0.08 0.76 0.24 1.0
## voc         0.81 -0.01  0.07 0.69 0.31 1.0
## addition    0.04  0.74 -0.21 0.50 0.50 1.2
## compterp   -0.05  0.72  0.05 0.53 0.47 1.0
## discrim     0.01  0.48  0.34 0.46 0.54 1.8
##
##                        ML1  ML2  ML3
## SS loadings           2.21 1.33 1.32
## Proportion Var        0.25 0.15 0.15
## Cumulative Var        0.25 0.39 0.54
## Proportion Explained  0.46 0.27 0.27
## Cumulative Proportion 0.46 0.73 1.00
##
##  With factor correlations of
##      ML1  ML2  ML3
## ML1 1.00 0.26 0.39
## ML2 0.26 1.00 0.35
## ML3 0.39 0.35 1.00
##
## Mean item complexity =  1.1
## Test of the hypothesis that 3 factors are sufficient.
##
```

```
## The degrees of freedom for the null model are  36  and the objective function was  3.05 with Chi Squa
## The degrees of freedom for the model are 12  and the objective function was  0.08
##
## The root mean square of the residuals (RMSR) is  0.02
## The df corrected root mean square of the residuals is  0.03
##
## The harmonic number of observations is  301 with the empirical chi square  8.03  with prob <  0.78
## The total number of observations was  301  with Likelihood Chi Square =  22.38  with prob <  0.034
##
## Tucker Lewis Index of factoring reliability =  0.964
## RMSEA index =  0.055  and the 90 % confidence intervals are  0.015 0.088
## BIC =  -46.11
## Fit based upon off diagonal values = 1
## Measures of factor score adequacy
##                                                 ML1  ML2  ML3
## Correlation of scores with factors             0.94 0.86 0.85
## Multiple R square of scores with factors       0.89 0.74 0.72
## Minimum correlation of possible factor scores  0.78 0.47 0.44
```

Comme on analyse ici directement le tableau des observations, on peut calculer les *scores factoriels*. En ACP, il n'y a qu'une seule manière de calculer les scores aux composantes et il suffit d'utiliser l'argument `scores = TRUE`. En AFC, il existe plusieurs algorithmes, le plus courant utilisant une méthode de régression. C'est celle-ci qui est utilisée dans l'exemple qui suit :

```
holzafc <- fa (holz [,5:13], nfactors = 3, fm = "ml", rotate = "promax",
               scores = "regression")
```

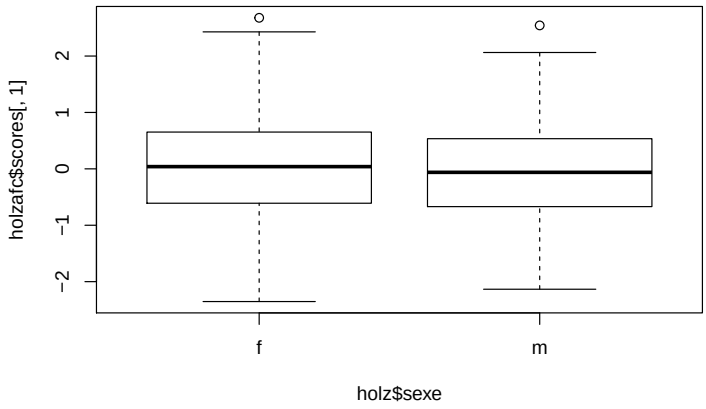
L'objet-résultat `holzafc` comprend cette fois 49 éléments. Les scores factoriels sont stockés dans l'élément `scores` qui est lui-même un tableau de données comprenant 301 lignes (les 301 enfants) et trois colonnes (correspondant aux trois facteurs). On visualise ici les premières lignes de ce tableau (les scores factoriels sont standardisés) :

```
head (holzafc$scores)
```

```
##              ML1        ML2        ML3
## [1,] -0.05272369 -0.1152457 -0.6845083
## [2,] -1.00054444  0.7432795  0.3868834
## [3,] -1.89170574 -1.3242455 -0.7744603
## [4,]  0.01330515 -0.6272115  0.6060389
## [5,] -0.12497752  0.2376895 -0.4699194
## [6,] -1.30587496  0.9936892  0.2149919
```

Ces scores peuvent ensuite être utilisés comme n'importe quelle autre variable. Par exemple, on peut se demander s'il existe des différences d'aptitudes en vocabulaire entre les garçons et les filles (1^{er} facteur : "ML1"). Les résultats suivants montrent qu'il n'y a pas de différences significatives.

```
plot (holzafc$scores[,1] ~ holz$sexe)
```




```
t.test (holzafc$scores[,1] ~ holz$sexe)
```

```
##
##  Welch Two Sample t-test
##
## data:  holzafc$scores[, 1] by holz$sexe
## t = 1.1518, df = 298.8, p-value = 0.2503
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -0.08859099  0.33866013
## sample estimates:
## mean in group f mean in group m
##      0.06064800     -0.06438657
```

Chapter 10

Analyse d'items

Le fichier "alter_ego.csv" (téléchargement) comprend les réponses de 182 sujets aux 12 items de la sous-échelle "cordialité - attitude amicale" de la dimension "Amabilité" du questionnaire de personnalité *Alter Ego*. Les réponses sont enregistrées sur une échelle de Likert en 5 points (de "tout à fait vrai"" à "tout à fait faux"). Il n'y a pas de données manquantes et tous les items ont été recodés dans le sens de l'amabilité.

Les 12 items :

- 16. Il n'est pas nécessaire de se comporter de façon cordiale avec tout le monde.
- 22. J'aime bien me mêler aux gens.
- 40. Je n'hésite pas à critiquer les autres, surtout quand ils le méritent.
- 44. En toute circonstance il m'est facile d'admettre que je me suis trompé.
- 52. Habituellement j'ai une attitude cordiale même avec des personnes pour lesquelles j'éprouve une certaine antipathie.
- 65. Je n'aime pas les groupes trop nombreux.
- 74. Habituellement, je ne fais pas facilement confiance aux personnes.
- 88. Je me confie volontiers aux autres.
- 93. J'estime qu'en chacun de nous il y a quelque chose de bon.
- 108. Si nécessaire, je n'hésite pas à dire aux autres de se mêler de ce qui les regarde.
- 126. Généralement j'ai confiance dans les autres et dans leurs intentions.
- 128. Avec certaines personnes il ne faut pas être trop tolérant.

```
ae <- read.csv2("alter_ego.csv")
str (ae)
```

```
## 'data.frame':    182 obs. of  12 variables:
##  $ ae016: int  4 3 4 2 3 4 3 3 4 4 ...
##  $ ae022: int  5 3 1 5 4 2 3 3 5 4 ...
##  $ ae040: int  2 3 5 4 2 3 2 3 2 2 ...
##  $ ae044: int  3 3 4 2 4 4 1 3 1 3 ...
##  $ ae052: int  2 4 2 4 4 4 2 3 4 5 ...
##  $ ae065: int  2 3 1 4 4 4 4 2 4 1 ...
##  $ ae074: int  1 2 2 4 2 2 2 3 2 3 ...
##  $ ae088: int  1 3 1 2 4 2 3 2 1 3 ...
##  $ ae093: int  4 5 5 5 4 4 3 3 5 5 ...
```

```
##  $ ae108: int  2 2 1 3 4 3 3 2 2 2 ...
##  $ ae126: int  2 4 3 3 4 3 3 3 2 4 ...
##  $ ae128: int  4 3 3 2 4 2 4 4 4 3 ...
```

L'analyse d'items repose sur le calcul de deux indices : la moyenne de l'item et l'indice de discrimination (r_{it}) qui sont fournis, avec d'autres résultats, par la fonction `alpha` du package `psych` (Revelle (2017)). Dans les commandes suivantes[1], les résultats sont stockés dans l'objet `fid_ae` :

```
library (psych)
fid_ae <- alpha (ae[,1:12])
str (fid_ae)
```

```
## List of 13
##  $ total        :'data.frame':   1 obs. of  8 variables:
##   ..$ raw_alpha: num 0.517
##   ..$ std.alpha: num 0.522
##   ..$ G6(smc)  : num 0.586
##   ..$ average_r: num 0.0835
##   ..$ S/N      : num 1.09
##   ..$ ase      : num 0.053
##   ..$ mean     : num 3.14
##   ..$ sd       : num 0.439
##  $ alpha.drop   :'data.frame':   12 obs. of  6 variables:
##   ..$ raw_alpha: num [1:12] 0.487 0.475 0.512 0.518 0.468 ...
##   ..$ std.alpha: num [1:12] 0.49 0.484 0.518 0.523 0.472 ...
##   ..$ G6(smc)  : num [1:12] 0.55 0.541 0.57 0.587 0.54 ...
##   ..$ average_r: num [1:12] 0.0805 0.0787 0.0889 0.0906 0.075 ...
##   ..$ S/N      : num [1:12] 0.962 0.94 1.073 1.096 0.892 ...
##   ..$ alpha se : num [1:12] 0.0563 0.0578 0.0535 0.0532 0.0586 ...
##  $ item.stats   :'data.frame':   12 obs. of  7 variables:
##   ..$ n     : num [1:12] 182 182 182 182 182 182 182 182 182 182 ...
##   ..$ raw.r : num [1:12] 0.418 0.462 0.33 0.312 0.487 ...
##   ..$ std.r : num [1:12] 0.434 0.454 0.338 0.318 0.496 ...
##   ..$ r.cor : num [1:12] 0.357 0.391 0.226 0.161 0.425 ...
##   ..$ r.drop: num [1:12] 0.235 0.282 0.132 0.11 0.309 ...
##   ..$ mean  : num [1:12] 3.23 3.54 2.76 3.26 3.37 ...
##   ..$ sd    : num [1:12] 1.05 1.06 1.08 1.09 1.07 ...
##  $ response.freq: num [1:12, 1:6] 0.0495 0.0275 0.0934 0.0549 0.0495 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:12] "ae016" "ae022" "ae040" "ae044" ...
##   .. ..$ : chr [1:6] "1" "2" "3" "4" ...
##  $ keys         : num [1:12] 1 1 1 1 1 1 1 1 1 1 ...
##  $ scores       : num [1:182] 2.67 3.17 2.67 3.33 3.58 ...
##  $ nvar         : int 12
##  $ boot.ci      : NULL
##  $ boot         : NULL
##  $ Unidim       :List of 1
##   ..$ Unidim: num 0.189
##  $ Fit          :List of 1
##   ..$ Fit.off: num 0.981
##  $ call         : language alpha(x = ae[, 1:12])
##  $ title        : NULL
```

[1] Il n'aurait pas été nécessaire ici d'indiquer les numéros de colonnes du tableau de données car celui-ci ne contient que les 12 items devant faire l'objet de l'analyse ; ce n'est pas le cas général et le plus souvent, le tableau de données contient plus de variables que celles devant faire l'objet d'une analyse d'items.

```
##  - attr(*, "class")= chr [1:2] "psych" "alpha"
```

L'objet-résultat est une liste comprenant 13 éléments. Parmi ceux-ci, les plus importants sont :

- `total` : différentes statistiques au niveau de l'échelle complète et notamment le coefficient *alpha* de Cronbach ;
- `item.stats` : différentes statistiques au niveau des items et notamment la moyenne (*mean*) et le coefficient de discrimination (r_{it}) (celui-ci est appelé *r.drop* dans le tableau de résultats) ;
- `response.freq` : la distribution des réponses en pourcentages pour chaque item.

```
fid_ae$total  # les statistiques de l'échelle
```

```
##  raw_alpha std.alpha   G6(smc) average_r      S/N        ase     mean
##   0.516841 0.5221298 0.5859169  0.083453 1.092618 0.05301325 3.141941
##         sd
##  0.439214
```

```
fid_ae$item.stats # les statistiques des items
```

```
##         n     raw.r     std.r     r.cor     r.drop     mean       sd
## ae016 182 0.4183480 0.4341077 0.3567103 0.23543085 3.225275 1.045283
## ae022 182 0.4621553 0.4541635 0.3906244 0.28150762 3.543956 1.064620
## ae040 182 0.3304561 0.3377162 0.2260362 0.13199505 2.763736 1.079289
## ae044 182 0.3116694 0.3178639 0.1607840 0.10978269 3.258242 1.089535
## ae052 182 0.4868338 0.4961642 0.4249285 0.30932668 3.373626 1.068363
## ae065 182 0.3648552 0.3333653 0.2361708 0.13011517 2.873626 1.279013
## ae074 182 0.5015905 0.4930738 0.4762920 0.31102570 3.000000 1.151506
## ae088 182 0.4295927 0.4178770 0.3264857 0.23122076 2.747253 1.133063
## ae093 182 0.2919889 0.3001371 0.1447520 0.09144935 4.291209 1.076022
## ae108 182 0.2673398 0.2731080 0.1207409 0.05898085 2.626374 1.108963
## ae126 182 0.5303918 0.5373040 0.5172568 0.36796406 3.181319 1.027297
## ae128 182 0.3985273 0.4025976 0.2914580 0.20266860 2.818681 1.100016
```

```
fid_ae$response.freq # la distribution des réponses aux items
```

```
##                1          2          3         4          5 miss
## ae016 0.04945055 0.21978022 0.27472527 0.3681319 0.08791209    0
## ae022 0.02747253 0.14835165 0.28571429 0.3296703 0.20879121    0
## ae040 0.09340659 0.37362637 0.28021978 0.1813187 0.07142857    0
## ae044 0.05494505 0.22527473 0.23076923 0.3846154 0.10439560    0
## ae052 0.04945055 0.19230769 0.20329670 0.4450549 0.10989011    0
## ae065 0.14285714 0.32417582 0.18131868 0.2197802 0.13186813    0
## ae074 0.08791209 0.30219780 0.22527473 0.2912088 0.09340659    0
## ae088 0.10989011 0.41208791 0.15384615 0.2692308 0.05494505    0
## ae093 0.05494505 0.03296703 0.04395604 0.3021978 0.56593407    0
## ae108 0.14835165 0.38461538 0.19780220 0.2307692 0.03846154    0
## ae126 0.07142857 0.18131868 0.29670330 0.3956044 0.05494505    0
## ae128 0.08241758 0.38461538 0.24175824 0.2142857 0.07692308    0
```

En indiquant uniquement le nom de l'objet-résultat, on obtient directement ces trois éléments, dans une présentation un peu différente :

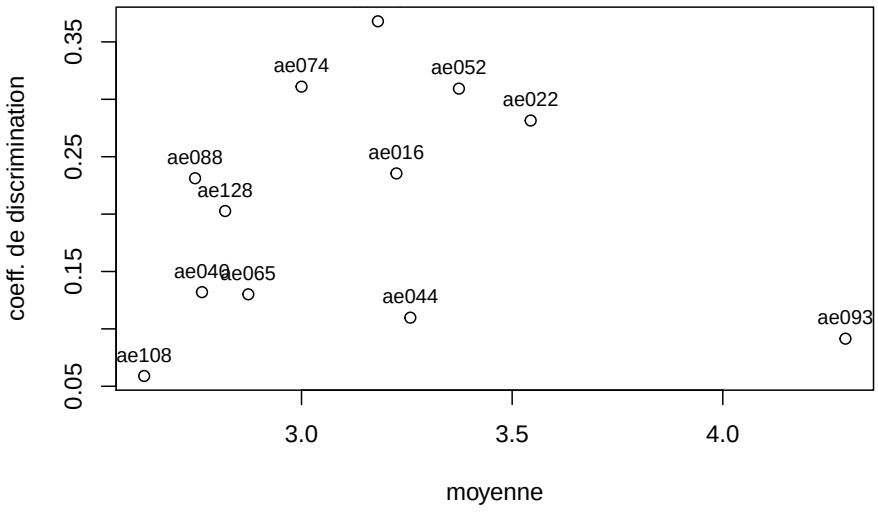
```
fid_ae
```

```
##
```

```
## Reliability analysis
## Call: alpha(x = ae[, 1:12])
##
##   raw_alpha std.alpha G6(smc) average_r S/N   ase mean   sd
##       0.52      0.52    0.59     0.083 1.1 0.053  3.1 0.44
##
##  lower alpha upper     95% confidence boundaries
## 0.41 0.52 0.62
##
##  Reliability if an item is dropped:
##       raw_alpha std.alpha G6(smc) average_r  S/N alpha se
## ae016      0.49      0.49    0.55     0.080 0.96    0.056
## ae022      0.47      0.48    0.54     0.079 0.94    0.058
## ae040      0.51      0.52    0.57     0.089 1.07    0.054
## ae044      0.52      0.52    0.59     0.091 1.10    0.053
## ae052      0.47      0.47    0.54     0.075 0.89    0.059
## ae065      0.52      0.52    0.56     0.089 1.08    0.053
## ae074      0.46      0.47    0.52     0.075 0.90    0.059
## ae088      0.49      0.50    0.56     0.082 0.98    0.057
## ae093      0.52      0.53    0.59     0.092 1.12    0.053
## ae108      0.53      0.53    0.59     0.095 1.15    0.052
## ae126      0.45      0.46    0.52     0.071 0.85    0.060
## ae128      0.49      0.50    0.56     0.083 1.00    0.056
##
##  Item statistics
##         n raw.r std.r r.cor r.drop mean  sd
## ae016 182  0.42  0.43  0.36  0.235  3.2 1.0
## ae022 182  0.46  0.45  0.39  0.282  3.5 1.1
## ae040 182  0.33  0.34  0.23  0.132  2.8 1.1
## ae044 182  0.31  0.32  0.16  0.110  3.3 1.1
## ae052 182  0.49  0.50  0.42  0.309  3.4 1.1
## ae065 182  0.36  0.33  0.24  0.130  2.9 1.3
## ae074 182  0.50  0.49  0.48  0.311  3.0 1.2
## ae088 182  0.43  0.42  0.33  0.231  2.7 1.1
## ae093 182  0.29  0.30  0.14  0.091  4.3 1.1
## ae108 182  0.27  0.27  0.12  0.059  2.6 1.1
## ae126 182  0.53  0.54  0.52  0.368  3.2 1.0
## ae128 182  0.40  0.40  0.29  0.203  2.8 1.1
##
## Non missing response frequency for each item
##          1    2    3    4    5 miss
## ae016 0.05 0.22 0.27 0.37 0.09    0
## ae022 0.03 0.15 0.29 0.33 0.21    0
## ae040 0.09 0.37 0.28 0.18 0.07    0
## ae044 0.05 0.23 0.23 0.38 0.10    0
## ae052 0.05 0.19 0.20 0.45 0.11    0
## ae065 0.14 0.32 0.18 0.22 0.13    0
## ae074 0.09 0.30 0.23 0.29 0.09    0
## ae088 0.11 0.41 0.15 0.27 0.05    0
## ae093 0.05 0.03 0.04 0.30 0.57    0
## ae108 0.15 0.38 0.20 0.23 0.04    0
## ae126 0.07 0.18 0.30 0.40 0.05    0
## ae128 0.08 0.38 0.24 0.21 0.08    0
```

On peut représenter graphiquement les deux coefficients centraux, la moyenne et le coefficient de discrimination avec les commandes suivantes[2] :

```
plot (fid_ae$item.stats[,6], fid_ae$item.stats[,5], xlab = "moyenne",
      ylab = "coeff. de discrimination")
text (fid_ae$item.stats[,6], fid_ae$item.stats[,5], names (ae[,1:12]), pos = 3, cex = 0.8)
```



[2]L'argument `cex` de la fonction `text` permet de réduire la taille de la police (ici à 80% de la taille normale).

Chapter 11

Analyse de variance

L'analyse de variance est conçue, dans `R`, comme un cas particulier du modèle linéaire qui intègre aussi notamment la régression multiple. La fonction générique est donc `lm` (pour *Linear Models*). La différence traditionnelle entre la régression multiple et l'analyse de variance tient au niveau de mesure des variables :

- dans la régression, les variables indépendantes (VI) sont numériques ;
- dans l'analyse de variance, les variables indépendantes sont des facteurs.

Il est très important, avant de mettre en oeuvre un modèle linéaire, de savoir (et de décider) si une variable est de nature numérique ou qualitative. La fonction `lm` ne gère pas l'analyse de la même manière dans le premier cas que dans le second.

11.1 Les objets facteurs et les objets numériques

Dans le fichier "notes250.csv" (téléchargement), la variable "sexe" est à l'origine reconnue comme une variable numérique. La fonction `str`[1] permet d'avoir des informations sur le type des éléments d'un objet. On peut aussi utiliser la fonction `is.factor` pour savoir directement si une variable est un facteur ou non :

```
n250 <- read.csv2("notes250.csv")
str (n250)
```

```
## 'data.frame':    250 obs. of  8 variables:
##  $ num   : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ sexe  : int  2 1 2 2 1 2 1 2 2 2 ...
##  $ cours1: int  11 -8 9 9 10 10 11 13 7 10 ...
##  $ cours2: int  13 14 12 12 10 12 13 10 13 11 ...
##  $ cours3: int  11 13 9 11 -8 9 9 6 4 12 ...
##  $ cours4: int  15 13 9 13 10 8 10 5 4 2 ...
##  $ cours5: int  13 -9 11 -9 8 -9 14 -9 11 -9 ...
##  $ cours6: int  -9 -8 -9 11 -9 9 -9 7 -9 6 ...
```

```
is.factor (n250$sexe)
```

```
## [1] FALSE
```

Définir une variable comme qualitative (facteur) s'effectue avec la fonction `factor`. Dans l'exemple qui suit, on créé une nouvelle variable dans le tableau de données ("sexef") définie comme un facteur :

[1] Cette fonction indique "int" pour la variable "sexe", l'abréviation d'"*integer*", c'est-à-dire "numérique".

```
n250$sexef <- factor (n250$sexe)
is.factor (n250$sexef)
```

```
## [1] TRUE
```

```
str (n250$sexef)
```

```
##  Factor w/ 2 levels "1","2": 2 1 2 2 1 2 1 2 2 2 ...
```

Etant donnée l'importance du modèle linéaire et de ses dérivés (GLM : *Generalized Linear Models* ; GLMM : *Generalized Linear Mixed Models*) dans `R` et plus généralement dans les techniques d'analyses statistiques contemporaines, on présentera une introduction à l'utilisation de la fonction `lm` dans les pages qui suivent. Toutefois, l'analyse de variance est sans doute plus simple à effectuer à partir du paquet `ez` (Lawrence (2016)), surtout pour des néophytes. La manière de réaliser des analyses de variance avec ce paquet sera donc aussi illustrée.

11.2 Analyse de variance avec une VI inter-individuelle

11.2.1 Les données

Dans l'exemple simple ci-dessous, on définit un tableau de données fictives (auquel on donne le nom `unevi`) avec 20 sujets et deux variables, une variable indépendante ("VI") et une variable dépendante ("VD") :

```
unevi <- data.frame (sujet = factor (1:20), VI = c (1,1,1,1,1,2,2,2,2,2,3,3,3,3,3,4,4,4,4,4),
                     VD = c(3,3,2,4,3,5,9,8,4,9,2,4,5,4,1,5,4,3,5,4))
unevi$VI <- as.factor (unevi$VI)
unevi
```

```
##    sujet VI VD
## 1      1  1  3
## 2      2  1  3
## 3      3  1  2
## 4      4  1  4
## 5      5  1  3
## 6      6  2  5
## 7      7  2  9
## 8      8  2  8
## 9      9  2  4
## 10    10  2  9
## 11    11  3  2
## 12    12  3  4
## 13    13  3  5
## 14    14  3  4
## 15    15  3  1
## 16    16  4  5
## 17    17  4  4
## 18    18  4  3
## 19    19  4  5
## 20    20  4  4
```

```
str (unevi)
```

```
## 'data.frame':    20 obs. of  3 variables:
##  $ sujet: Factor w/ 20 levels "1","2","3","4",..: 1 2 3 4 5 6 7 8 9 10 ...
##  $ VI   : Factor w/ 4 levels "1","2","3","4": 1 1 1 1 1 2 2 2 2 2 ...
```

```
##  $ VD   : num  3 3 2 4 3 5 9 8 4 9 ...
```

11.2.2 Statistiques descriptives

Les statistiques univariées de la variable dépendante en fonction des différentes modalités de la variable indépendante peuvent être calculées à partir des fonctions de base de `R`. Par exemple :

```
mean (unevi [unevi$VI == 1, "VD"])  # moyenne pour le groupe 1
```

```
## [1] 3
```

```
mean (unevi [unevi$VI == 2, "VD"])  # moyenne pour le groupe 2
```

```
## [1] 7
```

```
mean (unevi [unevi$VI == 3, "VD"])  # moyenne pour le groupe 3
```

```
## [1] 3.2
```

```
mean (unevi [unevi$VI == 4, "VD"])  # moyenne pour le groupe 4
```

```
## [1] 4.2
```

Cette manière de procéder est un peu fastidieuse et elle peut être réduite à une ligne de commande avec la fonction `tapply` :

```
tapply (unevi$VD, unevi$VI, mean)
```

```
##   1   2   3   4
## 3.0 7.0 3.2 4.2
```

Mais la fonction `tapply` doit être utilisée une seconde fois si on veut l'écart type, une troisième fois si on veut le nombre de sujets dans chacun des groupes, etc.

C'est pourquoi il est plus pratique d'utiliser une fonction qui calcule automatiquement plusieurs statistiques univariées pour chacun des groupes. C'est ce que fait la fonction `describeBy`[2] du paquet `psych`[3] (Revelle (2017)).

```
library (psych)
describeBy (unevi$VD, unevi$VI)
```

```
##
##  Descriptive statistics by group
## group: 1
##    vars n mean   sd median trimmed mad min max range skew kurtosis   se
## X1    1 5    3 0.71      3       3   0   2   4     2    0     -1.4 0.32
## ------------------------------------------------------------
## group: 2
##    vars n mean   sd median trimmed  mad min max range  skew kurtosis   se
## X1    1 5    7 2.35      8       7 1.48   4   9     5 -0.28    -2.14 1.05
## ------------------------------------------------------------
```

[2] Noter le *B* majuscule.
[3] Une alternative moins complète est disponible dans la fonction `brkdn` du paquet `prettyR` (Lemon & Grosjean (2015)).

```
## group: 3
##    vars n mean   sd median trimmed  mad min max range  skew kurtosis   se
## X1    1 5  3.2 1.64      4     3.2 1.48   1   5     4 -0.25    -1.99 0.73
## ------------------------------------------------------------
## group: 4
##    vars n mean   sd median trimmed  mad min max range  skew kurtosis   se
## X1    1 5  4.2 0.84      4     4.2 1.48   3   5     2 -0.25    -1.82 0.37
```

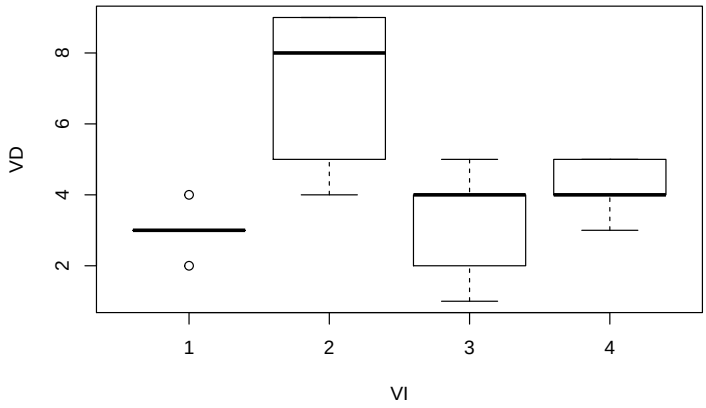
Dans la fonction `describeBy`, l'argument `mat = T` permet une présentation plus compacte des résultats[4] :

```
describeBy (unevi$VD, unevi$VI, mat = T)
```

```
##     item group1 vars n mean        sd median trimmed    mad min max range
## X11    1      1    1 5  3.0 0.7071068      3     3.0 0.0000   2   4     2
## X12    2      2    1 5  7.0 2.3452079      8     7.0 1.4826   4   9     5
## X13    3      3    1 5  3.2 1.6431677      4     3.2 1.4826   1   5     4
## X14    4      4    1 5  4.2 0.8366600      4     4.2 1.4826   3   5     2
##           skew  kurtosis        se
## X11  0.0000000 -1.400000 0.3162278
## X12 -0.2790991 -2.140496 1.0488088
## X13 -0.2488419 -1.989959 0.7348469
## X14 -0.2458756 -1.817959 0.3741657
```

Par défaut, la fonction `plot` fournit des boîtes de dispersion (médiane, intervalle inter-quartiles, intervalle de confiance autour de la médiane et points extrêmes) :

```
plot (VD~VI, data = unevi)
```



Pour obtenir un graphique avec les moyennes plutôt que les médianes, voir plus loin la présentation de l'analyse de variance avec le paquet `ez`.

11.2.3 La fonction `lm`

Les informations fournies directement par la fonction `lm` ne sont que des statistiques descriptives :

```
lm (VD~VI, data = unevi)
```

[4] Un autre avantage de cette option est qu'elle fournit les résultats sous forme d'un objet de type tableau de données (*data frame*), alors que la commande précédente fournit les résultats sous forme d'un objet de type liste

```
##
## Call:
## lm(formula = VD ~ VI, data = unevi)
##
## Coefficients:
## (Intercept)          VI2          VI3          VI4
##         3.0          4.0          0.2          1.2
```

Le premier de ces coefficients s'interprète comme la moyenne du premier groupe (groupe de référence) et les suivants comme les différences de moyennes avec le premier groupe.

Il faut appliquer la fonction `anova` sur les résultats précédents pour obtenir le tableau d'analyse de variance traditionnel :

```
anova (lm (VD~VI, data = unevi))
```

```
## Analysis of Variance Table
##
## Response: VD
##           Df Sum Sq Mean Sq F value   Pr(>F)
## VI         3  50.95  16.983   7.227 0.002782 **
## Residuals 16  37.60   2.350
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Appliquée à l'objet qui résulte de la fonction `lm`, la fonction `summary` fournit les résultats sous forme de tableau de régression. On verra dans le chapitre consacré à la régression comment interpréter ces différents éléments. Pour l'instant, on se contentera de relever que cette fontion permet d'obtenir le pourcentage de variance expliquée sous l'appellation "Multiple R-squared" (et qui correspond, en analyse de variance au coefficient que l'on appelle généralement coefficient de corrélation intra-classe ou eta^2) :

```
summary (lm (VD~VI, data = unevi))
```

```
##
## Call:
## lm(formula = VD ~ VI, data = unevi)
##
## Residuals:
##    Min     1Q Median     3Q    Max
##  -3.00  -1.05   0.00   0.85   2.00
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   3.0000     0.6856   4.376 0.000470 ***
## VI2           4.0000     0.9695   4.126 0.000793 ***
## VI3           0.2000     0.9695   0.206 0.839171
## VI4           1.2000     0.9695   1.238 0.233681
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.533 on 16 degrees of freedom
## Multiple R-squared:  0.5754, Adjusted R-squared:  0.4958
## F-statistic: 7.227 on 3 and 16 DF,  p-value: 0.002782
```

11.2.4 Le paquet ez

Dans le paquet `ez` (prononcer "*easy*"), la fonction de base est `ezANOVA`. Il est **indispensable** que le tableau de données comporte une variable identifiant les individus. Cette variable doit être définie comme un facteur. Dans ce tableau de données, cette variable s'appelle `sujet` :

```
str (unevi)
```

```
## 'data.frame':    20 obs. of  3 variables:
##  $ sujet: Factor w/ 20 levels "1","2","3","4",..: 1 2 3 4 5 6 7 8 9 10 ...
##  $ VI   : Factor w/ 4 levels "1","2","3","4": 1 1 1 1 1 2 2 2 2 2 ...
##  $ VD   : num  3 3 2 4 3 5 9 8 4 9 ...
```

Les principaux arguments de la fonction `ez` sont :

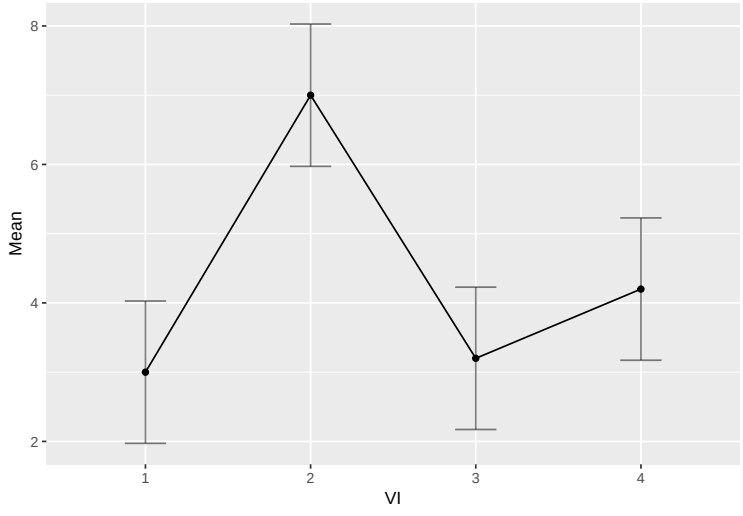
- `data` : le nom du tableau de données ;
- `dv` : le nom de la variable dépendante ;
- `between` : le nom de la variable indépendante inter-individuelle ;
- `wid` : le nom de la variable qui identifie les individus ;
- `detailed` : pour obtenir des résultats plus ou moins détaillés (et notamment le pourcentage de variance expliquée[5]).

```
library (ez)
ezANOVA(data = unevi, wid = sujet, dv = VD, between = VI, detailed = T)
```

```
## $ANOVA
##   Effect DFn DFd   SSn  SSd       F           p p<.05       ges
## 1     VI   3  16 50.95 37.6 7.22695 0.002782234     * 0.5753811
##
## $`Levene's Test for Homogeneity of Variance`
##   DFn DFd SSn SSd   F         p p<.05
## 1   3  16   6  20 1.6 0.2286475
```

Une représentation graphique des moyennes est obtenue avec la fonction `ezPlot` dont la structure est très proche de celle de la fonction précédente[6] :

```
ezPlot(data = unevi, wid = sujet, dv = VD, between = VI, x = VI)
```



[5] Dans les résultats, le pourcentage de variance expliquée est appellé "ges" (*generalized eta-squared*).

[6] Le paquet `ez` comporte aussi une fonction `ezStats` pour calculer des statistiques descriptives dont l'utilisation est très proche de celle des autres fonctions du paquet (`ezANOVA` et `ezPlot`).

11.2.5 Le paquet ez et les données manquantes

Le paquet `ez` s'avère une alternative très intéressante aux fonctions de base de `R` pour réaliser des analyses de variance et c'est tout particulièrement vrai pour les analyses comprenant des variables indépendantes intra-individuelles comme on le verra plus loin. L'un des inconvénients de ce paquet est de ne pas gérer du tout les données manquantes. Si le tableau de données comporte des données manquantes sur l'une ou l'autre des variables prises en compte dans les analyses (variables indépendantes ou dépendante), il faut prendre une décision les concernant avant l'analyse. Le plus souvent, même si c'est discutable du point de vue statistique, toutes les lignes comportant des données manquantes sont supprimées du tableau de données. Cela s'effectue facilement à l'aide de la fonction `na.omit`.

11.2.6 Comparaisons multiples

Les comparaisons multiples deux à deux entre les différents groupes peuvent être ajustées par la correction de *Bonferroni*[7] disponible dans la fonction `pairwise.t.test`:

```
pairwise.t.test (unevi$VD, unevi$VI, p.adj = "bonferroni")
```

```
##
##  Pairwise comparisons using t tests with pooled SD
##
## data:  unevi$VD and unevi$VI
##
##   1      2      3
## 2 0.0048 -      -
## 3 1.0000 0.0073 -
## 4 1.0000 0.0642 1.0000
##
## P value adjustment method: bonferroni
```

Le test de Tukey a sa propre fonction (`TukeyHSD`) :

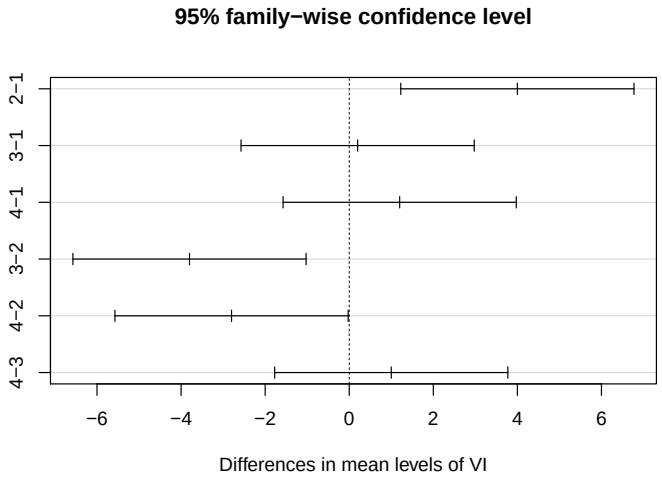
```
TukeyHSD (aov (VD~VI, data = unevi))
```

```
##   Tukey multiple comparisons of means
##     95% family-wise confidence level
##
## Fit: aov(formula = VD ~ VI, data = unevi)
##
## $VI
##     diff       lwr         upr     p adj
## 2-1  4.0  1.226138  6.77386163 0.0039662
## 3-1  0.2 -2.573862  2.97386163 0.9967637
## 4-1  1.2 -1.573862  3.97386163 0.6131097
## 3-2 -3.8 -6.573862 -1.02613837 0.0060368
## 4-2 -2.8 -5.573862 -0.02613837 0.0474772
## 4-3  1.0 -1.773862  3.77386163 0.7339787
```

Les résultats du test de Tukey peuvent être visualisés graphiquement avec la fonction générique `plot` :

```
plot (TukeyHSD (aov (VD~VI, data = unevi)))
```

[7] Généralement, on utilise la correction de *Bonferroni* lorsqu'il y a peu de comparaisons et le test de *Tukey* lorsque le nombre de comparaisons augmente. D'autres méthodes d'ajustement des comparaisons multiples sont disponibles, par exemple dans le paquet `multcomp` (Hothorn, Bretz, & Westfall (2008)).



11.3 Analyse de variance avec deux VI inter-individuelles

11.3.1 Les données

Les données fictives suivantes (n = 24) avec deux VI interindividuelles (R avec trois modalités et E avec deux modalités) composent le tableau de données `deuxvi`. Pour concrétiser l'exemple, on pourra penser à :

- VD = la performance à une tâche
- R : la quantité d'alcool ingérée (nulle, faible, importante)
- E : l'âge (jeunes vs. âgés)

```
deuxvi <- data.frame (sujet = 1:24,
                      R = c("r1","r1","r1","r1","r1","r1","r1","r1","r2",
                            "r2","r2","r2","r2","r2","r2","r2","r3","r3",
                            "r3","r3","r3","r3","r3","r3"),
                      E = c("e1","e1","e1","e1","e2","e2","e2","e2","e1",
                            "e1","e1","e1","e2","e2", "e2","e2","e1","e1",
                            "e1","e1","e2","e2","e2","e2"),
                      VD = c(26.2, 26.0, 25.0, 25.4, 24.8, 24.6, 26.7, 25.2,
                             25.7, 26.3,25.1,26.4, 19.6, 21.1, 19.0, 18.6,
                             22.8, 19.4, 18.8, 19.2, 19.8, 21.4, 22.8,21.3))
str(deuxvi)
```

```
## 'data.frame':    24 obs. of  4 variables:
##  $ sujet: int  1 2 3 4 5 6 7 8 9 10 ...
##  $ R    : Factor w/ 3 levels "r1","r2","r3": 1 1 1 1 1 1 1 1 2 2 ...
##  $ E    : Factor w/ 2 levels "e1","e2": 1 1 1 1 2 2 2 2 1 1 ...
##  $ VD   : num  26.2 26 25 25.4 24.8 24.6 26.7 25.2 25.7 26.3 ...
```

```
deuxvi
```

```
##    sujet  R  E   VD
## 1      1 r1 e1 26.2
## 2      2 r1 e1 26.0
## 3      3 r1 e1 25.0
## 4      4 r1 e1 25.4
## 5      5 r1 e2 24.8
```

```
## 6      6 r1 e2 24.6
## 7      7 r1 e2 26.7
## 8      8 r1 e2 25.2
## 9      9 r2 e1 25.7
## 10    10 r2 e1 26.3
## 11    11 r2 e1 25.1
## 12    12 r2 e1 26.4
## 13    13 r2 e2 19.6
## 14    14 r2 e2 21.1
## 15    15 r2 e2 19.0
## 16    16 r2 e2 18.6
## 17    17 r3 e1 22.8
## 18    18 r3 e1 19.4
## 19    19 r3 e1 18.8
## 20    20 r3 e1 19.2
## 21    21 r3 e2 19.8
## 22    22 r3 e2 21.4
## 23    23 r3 e2 22.8
## 24    24 r3 e2 21.3
```

11.3.2 Statistiques descriptives

Le plan expérimental est équilibré, comme le montre le résultat suivant :

```
table (deuxvi$R, deuxvi$E)
```

```
##
##      e1 e2
##   r1  4  4
##   r2  4  4
##   r3  4  4
```

Pour obtenir les moyennes avec la fonction `tapply`, il faut indiquer les deux variables indépendantes sous forme d'une liste :

```
tapply(deuxvi$VD, list(deuxvi$E, deuxvi$R), mean)
```

```
##        r1     r2     r3
## e1 25.650 25.875 20.050
## e2 25.325 19.575 21.325
```

Sinon, la fonction `describeBy` du paquet `psych` fournit directement des informations plus complètes :

```
describeBy(deuxvi$VD, list (deuxvi$E, deuxvi$R), mat = T)
```

```
##     item group1 group2 vars n   mean        sd median trimmed     mad  min
## X11    1     e1     r1    1 4 25.650 0.5507571  25.70  25.650 0.59304 25.0
## X12    2     e2     r1    1 4 25.325 0.9500000  25.00  25.325 0.44478 24.6
## X13    3     e1     r2    1 4 25.875 0.6020797  26.00  25.875 0.51891 25.1
## X14    4     e2     r2    1 4 19.575 1.0965856  19.30  19.575 0.74130 18.6
## X15    5     e1     r3    1 4 20.050 1.8502252  19.30  20.050 0.44478 18.8
## X16    6     e2     r3    1 4 21.325 1.2257651  21.35  21.325 1.11195 19.8
##      max range        skew  kurtosis        se
## X11 26.2   1.2 -0.12121178 -2.214973 0.2753785
## X12 26.7   2.1  0.60413326 -1.794681 0.4750000
## X13 26.4   1.3 -0.28564781 -2.105288 0.3010399
```

```
## X14 21.1   2.5  0.46062937 -1.889781 0.5482928
## X15 22.8   4.0  0.70868238 -1.712940 0.9251126
## X16 22.8   3.0 -0.04576245 -1.876868 0.6128825
```

On peut aussi demander les moyennes marginales (i.e. d'une seule VI, sans tenir compte de l'autre) :

```
describeBy(deuxvi$VD, deuxvi$R, mat = T)
```

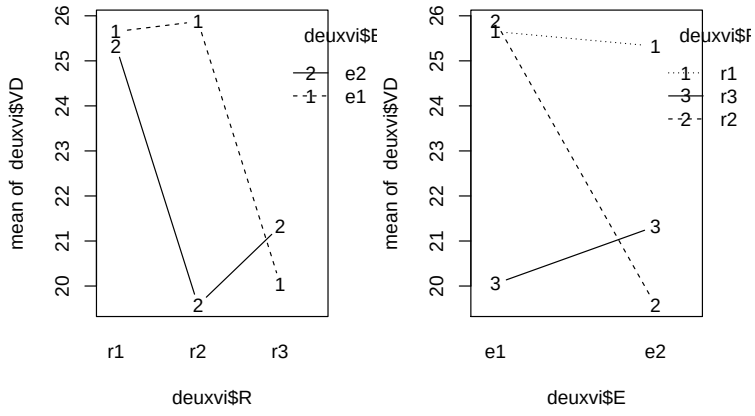
```
##     item group1 vars n    mean        sd median trimmed     mad  min  max
## X11    1     r1    1 8 25.4875 0.7395703  25.30 25.4875 0.88956 24.6 26.7
## X12    2     r2    1 8 22.7250 3.4656477  23.10 22.7250 4.81845 18.6 26.4
## X13    3     r3    1 8 20.6875 1.6048476  20.55 20.6875 1.85325 18.8 22.8
##     range        skew  kurtosis        se
## X11   2.1  0.34441984 -1.581392 0.2614776
## X12   7.8 -0.06496636 -2.098571 1.2252915
## X13   4.0  0.19950845 -1.847013 0.5673993
```

```
describeBy(deuxvi$VD, deuxvi$E, mat = T)
```

```
##     item group1 vars  n     mean       sd median trimmed     mad  min  max
## X11    1     e1    1 12 23.85833 3.005891  25.25   24.11 1.48260 18.8 26.4
## X12    2     e2    1 12 22.07500 2.702230  21.35   21.96 3.03933 18.6 26.7
##     range       skew  kurtosis        se
## X11   7.6 -0.7773946 -1.293767 0.8677259
## X12   8.1  0.2808046 -1.515245 0.7800665
```

Les représentations graphiques des moyennes sont obtenues ici avec la fonction `interaction.plot`. Les deux graphiques suivants illustrent la différence de représentation selon l'ordre dans lequel les VI sont indiquées :

```
par (mfrow = c(1,2))
interaction.plot (deuxvi$R, deuxvi$E, deuxvi$VD, type = "b")
interaction.plot (deuxvi$E, deuxvi$R, deuxvi$VD, type = "b")
```



11.3.3 L'analyse de variance avec la fonction `lm`

L'analyse de variance avec la fonction `lm` nécessite d'indiquer quels effets il faut introduire dans le modèle. La notation `VD ~ R + E + R:E` signifie que l'on introduit l'effet principal de "R", l'effet principal de "E" et l'interaction entre "R" et "E" (cette interaction est notée `R:E`)

```
anova (lm (VD ~ R + E + R:E, data = deuxvi))
```

```
## Analysis of Variance Table
##
## Response: VD
##           Df Sum Sq Mean Sq F value    Pr(>F)
## R          2 92.861  46.430  36.195 4.924e-07 ***
## E          1 19.082  19.082  14.875  0.001155 **
## R:E        2 63.761  31.880  24.853 6.635e-06 ***
## Residuals 18 23.090   1.283
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

La notation `VD ~ R + E + R:E` peut être simplifiée en `VD ~ R * E`. On obtient bien sûr les mêmes résultats :

```
anova (lm (VD ~ R * E, data = deuxvi))
```

```
## Analysis of Variance Table
##
## Response: VD
##           Df Sum Sq Mean Sq F value    Pr(>F)
## R          2 92.861  46.430  36.195 4.924e-07 ***
## E          1 19.082  19.082  14.875  0.001155 **
## R:E        2 63.761  31.880  24.853 6.635e-06 ***
## Residuals 18 23.090   1.283
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

11.3.4 L'analyse de variance avec le paquet ez

L'introduction de deux variables indépendantes avec la fonction `ezANOVA` comporte une particularité puisqu'il faut écrire : `between = .(R,E)` (noter bien le point).

Avant de réaliser l'analyse, il est prudent de vérifier que les variables indépendantes et la variable qui identifie les sujets sont bien définies comme des facteurs, ce qui n'est pas le cas ici pour la variable "sujet". Celle-ci est donc définie comme tel avant l'analyse :

```
str (deuxvi)
```

```
## 'data.frame':    24 obs. of  4 variables:
##  $ sujet: int  1 2 3 4 5 6 7 8 9 10 ...
##  $ R    : Factor w/ 3 levels "r1","r2","r3": 1 1 1 1 1 1 1 1 2 2 ...
##  $ E    : Factor w/ 2 levels "e1","e2": 1 1 1 1 2 2 2 2 1 1 ...
##  $ VD   : num  26.2 26 25 25.4 24.8 24.6 26.7 25.2 25.7 26.3 ...
```

```
ezANOVA(data = deuxvi, wid = sujet, dv = VD, between = .(R,E), detailed = T)
```

```
## Warning: Converting "sujet" to factor for ANOVA.
```

```
## $ANOVA
##   Effect DFn DFd      SSn   SSd        F            p p<.05       ges
## 1      R   2  18 92.86083 23.09 36.19521 4.924368e-07     * 0.8008639
## 2      E   1  18 19.08167 23.09 14.87527 1.155348e-03     * 0.4524760
## 3    R:E   2  18 63.76083 23.09 24.85264 6.635131e-06     * 0.7341419
##
## $`Levene's Test for Homogeneity of Variance`
##   DFn DFd      SSn   SSd         F         p p<.05
```

```
## 1   5  18 1.008333 13.65 0.2659341 0.9258477
```

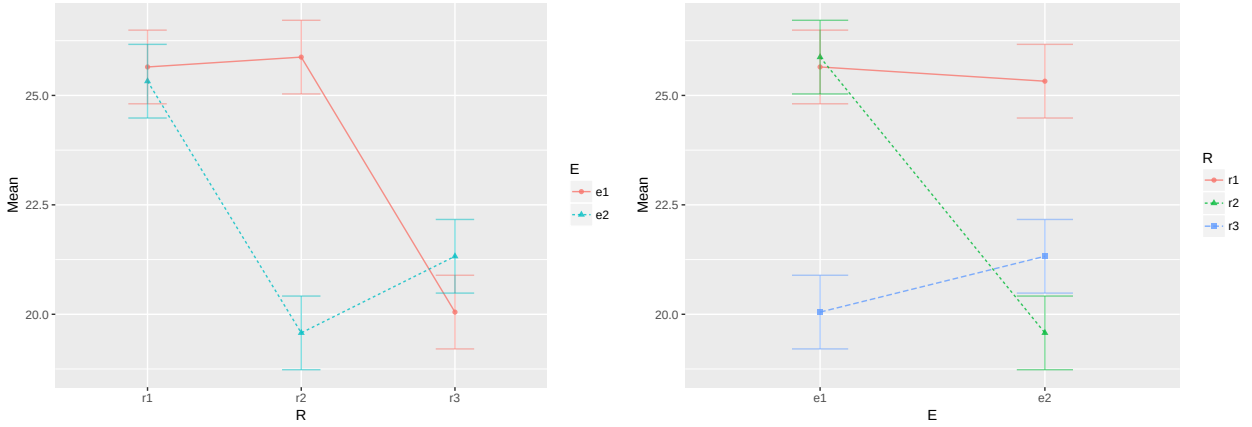
```
deuxvi$sujet <- factor (deuxvi$sujet)
ezANOVA(data = deuxvi, wid = sujet, dv = VD, between = .(R,E), detailed = T)
```

```
## $ANOVA
##   Effect DFn DFd      SSn   SSd        F            p p<.05       ges
## 1      R   2  18 92.86083 23.09 36.19521 4.924368e-07     * 0.8008639
## 2      E   1  18 19.08167 23.09 14.87527 1.155348e-03     * 0.4524760
## 3    R:E   2  18 63.76083 23.09 24.85264 6.635131e-06     * 0.7341419
##
## $`Levene's Test for Homogeneity of Variance`
##   DFn DFd      SSn   SSd         F         p p<.05
## 1   5  18 1.008333 13.65 0.2659341 0.9258477
```

La représentation graphique des moyennes obtenue avec `ezPlot` est plus esthétique que celle fournie par `interaction.plot`. L'argument `x` indique quelle variable va figurer sur l'axe des abscisses ; l'argument `split` indique l'autre VI et l'argument `do_lines` permet de tracer des lignes entre les différents points d'une même modalité) :

Les deux graphiques suivants se différencient par la variable mentionnée sur l'axe des abscisses :

```
ezPlot(data = deuxvi, wid = sujet, dv = VD, between = .(R,E), x = R, split = E, do_lines = T)
ezPlot(data = deuxvi, wid = sujet, dv = VD, between = .(R,E), x = E, split = R, do_lines = T)
```



11.3.5 Comparaisons multiples

Dans les analyses avec plusieurs variables indpéendantes, les comparaisons de moyenne deux à deux posent des problèmes statistiques assez compliqués. On examinera ici quelques stratégies possibles.

11.3.5.1 Les comparaisons avec des test de *Student*

La comparaison des moyennes des groupes avec le test *t de Student* est techniquement simple, mais elle est statistiquement contestable :

- la multiplication des comparaisons augmente la probabilité de trouver des différences significatives ;
- le test de Student est effectué en ne tenant compte que d'une partie de la variance globale (la somme des carrés résiduelle utilisée et son nombre de degré de liberté n'est donc pas celle de l'analyse de variance

globale[8]).

Voici quelques exemples quant à la manière de procéder.

Le premier exemple compare les modalités “r1” et “r2” de de la variable “R”, sans tenir compte de la troisième modalité, ni de la seconde variable indépendante. Les deux exemples suivants sont équivalentes, mais la seconde est un peu plus longue à écrire

```
t.test (deuxvi$VD[deuxvi$R == "r1"], deuxvi$VD[deuxvi$R == "r2"])
```

```
##
##  Welch Two Sample t-test
##
## data:  deuxvi$VD[deuxvi$R == "r1"] and deuxvi$VD[deuxvi$R == "r2"]
## t = 2.2049, df = 7.6362, p-value = 0.06011
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -0.1507803  5.6757803
## sample estimates:
## mean of x mean of y
##   25.4875   22.7250
```

```
t.test (deuxvi[deuxvi$R == "r1" | deuxvi$R == "r2", "VD"] ~
          deuxvi[deuxvi$R == "r1" | deuxvi$R == "r2", "R"])
```

```
##
##  Welch Two Sample t-test
##
## data:  deuxvi[deuxvi$R == "r1" | deuxvi$R == "r2", "VD"] by deuxvi[deuxvi$R == "r1" | deuxvi$R == "r
## t = 2.2049, df = 7.6362, p-value = 0.06011
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -0.1507803  5.6757803
## sample estimates:
## mean in group r1 mean in group r2
##          25.4875          22.7250
```

Le second exemple compare les moyennes du groupe composé des individus ayant les modalités “r1” et “e1” avec le groupe des individus ayant les modalités “r1” et “e2” :

```
t.test (deuxvi$VD[deuxvi$R == "r1" & deuxvi$E == "e1"],
        deuxvi$VD[deuxvi$R == "r1" & deuxvi$E == "e2"])
```

```
##
##  Welch Two Sample t-test
##
## data:  deuxvi$VD[deuxvi$R == "r1" & deuxvi$E == "e1"] and deuxvi$VD[deuxvi$R == "r1" & deuxvi$E == "
## t = 0.59193, df = 4.8119, p-value = 0.5806
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -1.103137  1.753137
## sample estimates:
## mean of x mean of y
##    25.650    25.325
```

[8] Voir ci-dessous le paragraphe sur les effets simples pour voir comment il est possible de contourner cette difficulté.

11.3.5.2 Les comparaisons avec le test de Tukey

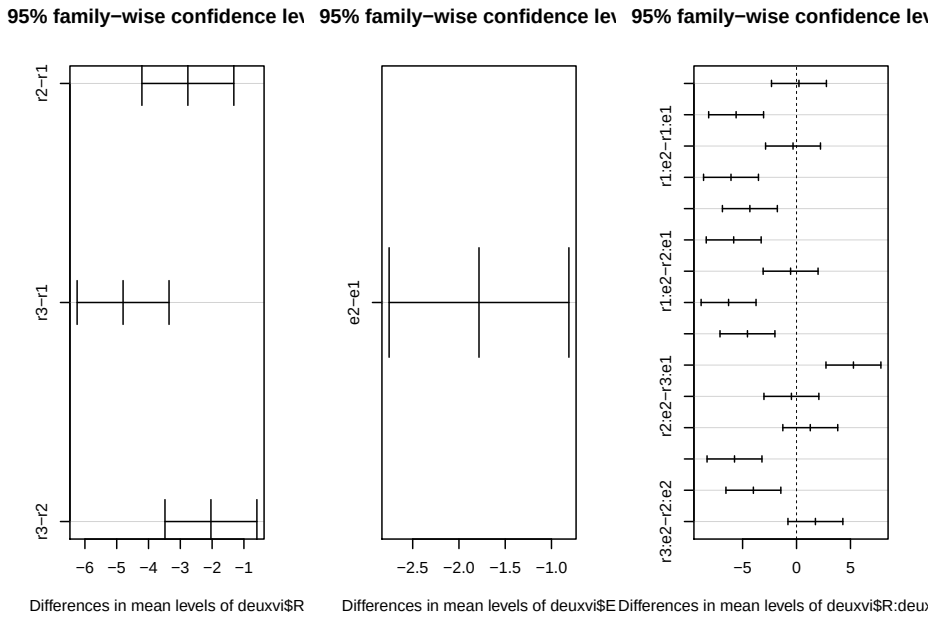
Comme dans le cas avec une VI inter-individuelle, le test de *Tukey* peut être utilisé pour réaliser toutes les comparaisons possibles entre les groupes. Cette méthode simple n'est pourtant pas la plus efficace du point de vue statistique, car le test manque de puissance lorsqu'il y a beaucoup de groupes[9] :

```
TukeyHSD (aov ((deuxvi$VD ~ deuxvi$R * deuxvi$E)))
```

```
##   Tukey multiple comparisons of means
##     95% family-wise confidence level
##
## Fit: aov(formula = (deuxvi$VD ~ deuxvi$R * deuxvi$E))
##
## $`deuxvi$R`
##          diff       lwr        upr     p adj
## r2-r1 -2.7625 -4.207787 -1.3172128 0.0003395
## r3-r1 -4.8000 -6.245287 -3.3547128 0.0000003
## r3-r2 -2.0375 -3.482787 -0.5922128 0.0055472
##
## $`deuxvi$E`
##            diff      lwr        upr     p adj
## e2-e1 -1.783333 -2.75476 -0.8119067 0.0011553
##
## $`deuxvi$R:deuxvi$E`
##                diff        lwr       upr     p adj
## r2:e1-r1:e1  0.225 -2.3201851  2.770185 0.9997185
## r3:e1-r1:e1 -5.600 -8.1451851 -3.054815 0.0000204
## r1:e2-r1:e1 -0.325 -2.8701851  2.220185 0.9983324
## r2:e2-r1:e1 -6.075 -8.6201851 -3.529815 0.0000068
## r3:e2-r1:e1 -4.325 -6.8701851 -1.779815 0.0004825
## r3:e1-r2:e1 -5.825 -8.3701851 -3.279815 0.0000120
## r1:e2-r2:e1 -0.550 -3.0951851  1.995185 0.9811892
## r2:e2-r2:e1 -6.300 -8.8451851 -3.754815 0.0000041
## r3:e2-r2:e1 -4.550 -7.0951851 -2.004815 0.0002705
## r1:e2-r3:e1  5.275  2.7298149  7.820185 0.0000444
## r2:e2-r3:e1 -0.475 -3.0201851  2.070185 0.9902110
## r3:e2-r3:e1  1.275 -1.2701851  3.820185 0.6135909
## r2:e2-r1:e2 -5.750 -8.2951851 -3.204815 0.0000143
## r3:e2-r1:e2 -4.000 -6.5451851 -1.454815 0.0011258
## r3:e2-r2:e2  1.750 -0.7951851  4.295185 0.2914242
```

```
par (mfrow = c(1,3))
plot (TukeyHSD (aov ((deuxvi$VD ~ deuxvi$R * deuxvi$E))))
```

[9] C'est-à-dire qu'il faut des différences très importantes pour qu'elles soient significatives.



11.3.5.3 Les comparaisons avec des contrastes

Comparer des moyennes revient techniquement à tester la significativité de *contrastes*. Un contraste est une simple combinaison linéaire de moyennes. Appliqué à un objet de type `lm`, la fonction `summary` fournit le résultat d'une décomposition possible des différents effets en plusieurs contrastes indépendants[10].

```
summary (lm (deuxvi$VD ~ deuxvi$R * deuxvi$E))
```

```
##
## Call:
## lm(formula = deuxvi$VD ~ deuxvi$R * deuxvi$E)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.5250 -0.6687 -0.1500  0.4500  2.7500
##
## Coefficients:
##                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)            25.6500     0.5663  45.294  < 2e-16 ***
## deuxvi$Rr2              0.2250     0.8009   0.281    0.782
## deuxvi$Rr3             -5.6000     0.8009  -6.992 1.58e-06 ***
## deuxvi$Ee2             -0.3250     0.8009  -0.406    0.690
## deuxvi$Rr2:deuxvi$Ee2  -5.9750     1.1326  -5.275 5.13e-05 ***
## deuxvi$Rr3:deuxvi$Ee2   1.6000     1.1326   1.413    0.175
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.133 on 18 degrees of freedom
## Multiple R-squared:  0.8838, Adjusted R-squared:  0.8516
```

[10]Dans cet exemple avec deux VI comprenant respectivement trois et deux modalités, il y a deux contrastes relatifs à la première VI, un pour la seconde et deux pour l'interaction. De manière générale, si la première VI a p modalités et la seconde m modalités, il y a (p-1) contrastes pour tester l'effet de la première VI, (m-1) contrastes pour tester l'effet de la seconde et (p-1) × (m-1) contrastes pour tester l'effet de l'interaction.

```
## F-statistic: 27.39 on 5 and 18 DF,  p-value: 8.043e-08
```

Voyons la signification des différents coefficients de ce tableau de résultats :

```
##             Estimate Std. Error t value     Pr(>|t|)
## (Intercept)    25.65  0.5662989 45.2941 5.303226e-20
```

L'*intercept* est en fait simplement la moyenne du groupe de référence. Celui-ci est défini par défaut par la première modalité de la première VI et la première modalité de la seconde VI. Comme on peut le vérifier, il s'agit ici de la moyenne du groupe "r1e1".

```
##            Estimate Std. Error   t value  Pr(>|t|)
## deuxvi$Rr2    0.225  0.8008676 0.2809453 0.7819562
```

Le second coefficient revient à tester la différence entre le second groupe de la première VI et le groupe de référence de cette même VI, à l'intérieur de la modalité de référence de la seconde VI. Autrement dit, on teste ici la différence entre "e1r1" (groupe de référence) et "e1r2" (on peut en effet vérifier que : 25.875 - 25.650 = 0.2250).

```
##            Estimate Std. Error   t value     Pr(>|t|)
## deuxvi$Rr3     -5.6  0.8008676 -6.992417 1.576256e-06
```

Selon la même logique, le troisième coefficient permet de tester la différence entre "e1r1" (groupe de référence) et "e1r3" : 20.050 - 25.650 = -5.6)

```
##            Estimate Std. Error    t value  Pr(>|t|)
## deuxvi$Ee2   -0.325  0.8008676 -0.4058099 0.6896656
```

Le quatrième coefficient renvoie cette fois à la seconde variable indépendante et on teste la différence entre "e1r1" et "e2r1" (25.325 - 25.650 = -0.325)

```
##                         Estimate Std. Error   t value     Pr(>|t|)
## deuxvi$Rr2:deuxvi$Ee2   -5.975   1.132598 -5.275483 5.134385e-05
```

Ce cinquième coefficient est le premier contraste qui correspond à l'interaction et l'expression est un peu plus compliquée, même si la logique est similaire. On teste s'il y a une différence entre les modalités "r1" et "r2" de la variable "R" à l'intérieur de la première modalité de la variable "E" ("r1" *vs.* "r2" dans "e1") par rapport à la différence entre les mêmes modalités de la variable "R", mais cette fois pour la seconde modalité de la variable "E" ("r1" *vs.* "r2" dans "e2"). En abrégé : ("e2r2" - "e2r1") *vs.* ("e1r2" - "e1r1") ; Vérification : (19.575 - 25.875) - (25.325 - 25.650) = -5.975

```
##                         Estimate Std. Error  t value Pr(>|t|)
## deuxvi$Rr3:deuxvi$Ee2      1.6   1.132598 1.412682 0.174807
```

Enfin, le sixième coefficient correspond au second constraste de l'interaction : ("e2r3" - "e2r1") *vs.* ("e1r3" - "e1r1"). Vérification : (21.325 - 25.325) - (20.050 - 25.650) = 1.600

Il est possible de modifier automatiquement la liste des contrastes testés (par exemple en changeant le groupe de référence) ou en spécifiant manuellement les contrastes correspondant aux hypothèses théoriques qui ont présidé à l'expérience. La technicité de ce type d'opération dépasse le cadre de ce document et ne sera donc pas abordé ici.

11.3.5.4 La décomposition d'une interaction en effets simples

En psychologie, on trouve souvent une présentation des résultats d'une interaction sous la forme d'une décomposition en effets simples. Cela revient, dans l'exemple discuté :

- soit à comparer les différentes modalités de "E" à l'intérieur de chacune des modalités de "R" (c'est-à-dire "e1r1" *vs.* "e2r1" ; "e1r2" *vs.* "e2r2" ; "e1r3" *vs.* "e2r3") :

- soit à comparer les différentes modalités de "R" à l'intérieur de chacune des modalités de "E" (c'est-à-dire "r1e1" *vs.* "r2e1" ; "r1e1" *vs.* "r3e1" ; "r2e1" *vs.* "r3e1" ; "r1e1" *vs.* "r2e1" ; "r1e1" *vs.* "r3e1" ; "r2e1" *vs.* "r3e1")

Il ne semble pas exister de manière aisée et automatique permettant d'obtenir ces effets simples avec `R`. La solution proposée ici est un peu longue, parce qu'elle détaille tous les calculs nécessaires, mais elle fournit les résultats souhaités.

1ère étape : calculer et noter les moyennes des différentes groupes

```
mean (deuxvi$VD[deuxvi$R == "r1" & deuxvi$E == "e1"])
```

```
## [1] 25.65
```

```
mean (deuxvi$VD[deuxvi$R == "r2" & deuxvi$E == "e1"])
```

```
## [1] 25.875
```

```
mean (deuxvi$VD[deuxvi$R == "r3" & deuxvi$E == "e1"])
```

```
## [1] 20.05
```

```
mean (deuxvi$VD[deuxvi$R == "r1" & deuxvi$E == "e2"])
```

```
## [1] 25.325
```

```
mean (deuxvi$VD[deuxvi$R == "r2" & deuxvi$E == "e2"])
```

```
## [1] 19.575
```

```
mean (deuxvi$VD[deuxvi$R == "r3" & deuxvi$E == "e2"])
```

```
## [1] 21.325
```

2e étape : noter la valeur du carré moyen résiduel (ici : 1.283) de l'analyse de variance globale ainsi que ses degrés de liberté (ici : 18) :

```
anova (lm (deuxvi$VD ~ deuxvi$R * deuxvi$E))
```

```
## Analysis of Variance Table
##
## Response: deuxvi$VD
##                   Df Sum Sq Mean Sq F value    Pr(>F)
## deuxvi$R           2 92.861  46.430  36.195 4.924e-07 ***
## deuxvi$E           1 19.082  19.082  14.875  0.001155 **
## deuxvi$R:deuxvi$E  2 63.761  31.880  24.853 6.635e-06 ***
## Residuals         18 23.090   1.283
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

3e étape : calculer la probabilité associée aux différents contrastes souhaités en se servant de la fonction de distribution du *F de Fisher* (fonction `pf`) avec le carré moyen et le nombre de degrés de liberté de l'analyse de variance globale.

Par exemple, pour tester les effets simples de "E" dans "R", on effectuera les contrastes suivants[11]

- "e1" *vs.* "e2" dans "r1" (moyennes respectives de 25.65 et 25.325)

```
1 - pf ((((25.65 - 25.325)/sqrt (1.283))^2)*2,1,18)
```

```
## [1] 0.689691
```

[11] Il s'agit ici d'un test bilatéral. Pour obtenir les résultats d'un test unilatéral, il ne faut pas multiplier par 2 l'expression, soit la commande : `1 - pf (((mr1e1 - mr1e2)/sqrt (1.283))^2,1,18)`

- "e1" *vs.* "e2" dans "r2" (moyennes respectives de 25.875 et 19.575)

```
1 - pf ((((25.875 - 19.575)/sqrt (1.283))^2)*2,1,18)
```

```
## [1] 3.115815e-07
```

- "e1" *vs.* "e2" dans "r3" (moyennes respectives de 20.05 et 21.325)

```
1 - pf ((((20.05 - 21.325)/sqrt (1.283))^2)*2,1,18)
```

```
## [1] 0.1288193
```

Si on préfère tester les effets simples de "R" dans "E" , il faut procéder de la manière suivante[12] :

- "r1" *vs.* "r2" dans "e1"

```
1 - pf ((((25.65 - 25.875)/sqrt (1.283))^2)*2,1,18)
```

```
## [1] 0.7819746
```

- "r1" *vs.* "r3" dans "e1"

```
1 - pf ((((25.65 - 20.05)/sqrt (1.283))^2)*2,1,18)
```

```
## [1] 1.578085e-06
```

- "r2" *vs.* "r3" dans "e1"

```
1 - pf ((((25.875 - 20.05)/sqrt (1.283))^2)*2,1,18)
```

```
## [1] 9.263886e-07
```

- "r1" *vs.* "r2" dans "e2"

```
1 - pf ((((25.325 - 19.575)/sqrt (1.283))^2)*2,1,18)
```

```
## [1] 1.105069e-06
```

- "r1" *vs.* "r3" dans "e2"

```
1 - pf ((((25.325 - 21.325)/sqrt (1.283))^2)*2,1,18)
```

```
## [1] 9.402432e-05
```

- "r2" *vs.* "r3" dans "e2"

```
1 - pf ((((19.575 - 21.325)/sqrt (1.283))^2)*2,1,18)
```

```
## [1] 0.04235934
```

Une fonction pour calculer les effets simples

Comme la procédure précédente est un peu fastidieuse, il est possible d'écrire sa propre petite fonction qui permet d'automatiser ces différents calculs. La fonction s'appelle `ES` et elle est définie ci-dessous :

```
ES <- function (data, VI1, VI2, VD)
{
  k1 <- length (levels (data[,VI1]))
  k2 <- length (levels (data[,VI2]))
  lk1 <- rep (levels (data[,VI1]), each = k2*(k2-1)/2)
  x <- t (combn(levels (data[,VI2]), 2))
  x <- data.frame (VI1 = lk1, VI2_1 = rep (x[,1],k1), VI2_2 = rep (x[,2],k1))
```

[12] Comme il y a trois comparaisons pour chaque modalité de E, on pourrait, pour conclure à l'existence d'une différence statistiquement significative au seuil de 5%, appliquer la correction de Bonferroni, (ce qui revient à utiliser la probabilité de $0.05/3 = 0.01667$).

```
  y <- tapply(data[,VD], list(data[,VI1], data[,VI2]), mean)
  y <- data.frame (expand.grid (rownames (y), colnames (y)), as.numeric (y))
  colnames (y) <- c("VI1", "VI2_1", "moy_VI2_1")
  m <- merge (x,y, by.x = c ("VI1", "VI2_1"), by.y = c ("VI1", "VI2_1"))
  colnames (y) <- c("VI1", "VI2_2", "moy_VI2_2")
  m <- merge (m,y, by.x = c ("VI1", "VI2_2"), by.y = c ("VI1", "VI2_2"))
  m <- m [,  c("VI1", "VI2_1", "VI2_2", "moy_VI2_1", "moy_VI2_2")]
  x <- anova (lm (data[,VD] ~ data[,VI1] * data[,VI2]))
  m$p <- round (1 - pf ((((m$moy_VI2_1 - m$moy_VI2_2)/
                          sqrt (x ["Residuals","Mean Sq"]))^2)*2,1,x ["Residuals","Df"]),7)
  cat ("Effets simples de", VI2, "dans", VI1, "\n")
  m
}
```

Elle comporte quatre arguments qui permettent de préciser :

- le nom du tableau de données à analyser (argument `data`) ;
- le nom de la première variable indépendante (argument `VI1`) ;
- le nom de la seconde variable indépendante (argument `VI2`) ;
- le nom de la variable dépendante (argument `VD`).

Ainsi, pour obtenir les effets simples de R dans E, on écrira :

```
ES (data = deuxvi, VI1 = "E", VI2 = "R", VD = "VD")
```

```
## Effets simples de R dans E
```

```
##   VI1 VI2_1 VI2_2 moy_VI2_1 moy_VI2_2         p
## 1  e1    r1    r2    25.650    25.875 0.7819562
## 2  e1    r1    r3    25.650    20.050 0.0000016
## 3  e1    r2    r3    25.875    20.050 0.0000009
## 4  e2    r1    r2    25.325    19.575 0.0000011
## 5  e2    r1    r3    25.325    21.325 0.0000939
## 6  e2    r2    r3    19.575    21.325 0.0423434
```

Les résultats indiquent les moyennes comparées (“moy_VI2_1” et “moy_VI2_2”), les groupes correspondants ainsi que la probabilité associée au test de l’effet simple. Pour la première ligne ci-dessus, on compare donc les modalités “r1” (m = 25.650) et “r2” (25.875) dans la modalité “e1” de la variable “E”. La différence n’est pas statistiquement significative (p = 0.78).

Pour obtenir les effets simples de “E” dans “R”, il suffit de changer les noms des variables associées aux arguments `VI1` et `VI2`, comme ci-dessous :

```
ES (data = deuxvi, VI1 = "R", VI2 = "E", VD = "VD")
```

```
## Effets simples de E dans R
```

```
##   VI1 VI2_1 VI2_2 moy_VI2_1 moy_VI2_2         p
## 1  r1    e1    e2    25.650    25.325 0.6896656
## 2  r2    e1    e2    25.875    19.575 0.0000003
## 3  r3    e1    e2    20.050    21.325 0.1287883
```

11.4 Analyse de variance avec une VI intra-individuelle

Dans les plans avec des VI intraindividuelles, les mêmes sujets sont soumis aux différentes modalités de la VI. On parle aussi de plans à mesures répétées. Dans l’exemple ci-dessous, 12 sujets ont répété quatre fois de

suite la même tâche A chaque fois, on a compté le nombre d'erreurs. On cherche à savoir si l'entraînement a un effet sur la performance.

```
intra1 <- data.frame (sujet = factor (1:12),
                      e1 = c(18,19,14,16,12,18,16,18,16,19,16,16),
                      e2 = c(14,12,10,12,8,10,10,8,12,16,14,12),
                      e3 = c(12,8,6,10,6,5,8,4,6,10,10,8),
                      e4 = c(6,4,2,4,2,1,4,1,2,8,9,8))
intra1
```

```
##    sujet e1 e2 e3 e4
## 1      1 18 14 12  6
## 2      2 19 12  8  4
## 3      3 14 10  6  2
## 4      4 16 12 10  4
## 5      5 12  8  6  2
## 6      6 18 10  5  1
## 7      7 16 10  8  4
## 8      8 18  8  4  1
## 9      9 16 12  6  2
## 10    10 19 16 10  8
## 11    11 16 14 10  9
## 12    12 16 12  8  8
```

11.4.1 L'organisation des données pour l'analyse

La plupart des logiciels statistiques analysent des fichiers de données où chaque individu occupe une seule ligne comme le tableau de données de l'exemple ci-dessus. Pour les analyses de variance à mesures répétées, `R` requiert des tableaux de données dans lesquelles il n'y a qu'une seule observation par ligne. Il est possible grâce au paquet `reshape2` de passer facilement et automatiquement :

- d'un tableau de données avec une seule observation par ligne à un tableau avec une ligne par individu (fonction `dcast`) ;
- d'un tableau de données avec une seule ligne par individu à un tableau avec une seule observation par ligne (fonction `melt`).

C'est donc la fonction `melt` qu'il faut utiliser ici :

```
library(reshape2)
intra1m <- melt (intra1, id.vars = "sujet", measure.vars = c ("e1", "e2", "e3", "e4"),
                 variable.name = "essai", value.name = "erreurs")
intra1m
```

```
##    sujet essai erreurs
## 1      1    e1      18
## 2      2    e1      19
## 3      3    e1      14
## 4      4    e1      16
## 5      5    e1      12
## 6      6    e1      18
## 7      7    e1      16
## 8      8    e1      18
## 9      9    e1      16
## 10    10    e1      19
## 11    11    e1      16
## 12    12    e1      16
```

```
## 13     1    e2      14
## 14     2    e2      12
## 15     3    e2      10
## 16     4    e2      12
## 17     5    e2       8
## 18     6    e2      10
## 19     7    e2      10
## 20     8    e2       8
## 21     9    e2      12
## 22    10    e2      16
## 23    11    e2      14
## 24    12    e2      12
## 25     1    e3      12
## 26     2    e3       8
## 27     3    e3       6
## 28     4    e3      10
## 29     5    e3       6
## 30     6    e3       5
## 31     7    e3       8
## 32     8    e3       4
## 33     9    e3       6
## 34    10    e3      10
## 35    11    e3      10
## 36    12    e3       8
## 37     1    e4       6
## 38     2    e4       4
## 39     3    e4       2
## 40     4    e4       4
## 41     5    e4       2
## 42     6    e4       1
## 43     7    e4       4
## 44     8    e4       1
## 45     9    e4       2
## 46    10    e4       8
## 47    11    e4       9
## 48    12    e4       8
```

Le nouveau tableau de données (`intra1m`) contient donc 48 lignes (une par observation). Chaque individu est répété quatre fois, car il a été mesuré quatre fois. La variable dépendante s'appelle "erreurs" (elle a été nommée ainsi grâce à l'argument `value.name`) et les valeurs de cette variable sont fournies par les quatre variables "e1", "e2", "e3", "e4" de l'ancien tableau de données (argument `measure.vars`). La variable indépendante s'appelle "essai" (argument `variable.name`) et elle comporte quatre modalités ("e1", "e2", "e3", "e4").

11.4.2 Statistiques descriptives

Les statistiques descriptives sont toujours obtenues grâce à la fonction `describeBy` du paquet `psych` :

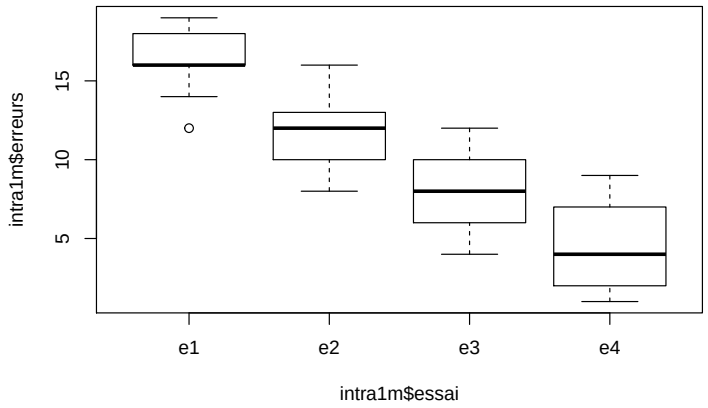
```
library (psych)
describeBy (intra1m$erreurs, intra1m$essai, mat = T)
```

```
##     item group1 vars  n  mean       sd median trimmed    mad min max range
## X11    1     e1    1 12 16.50 2.067058     16    16.7 2.9652  12  19     7
## X12    2     e2    1 12 11.50 2.430862     12    11.4 2.9652   8  16     8
```

```
## X13    3     e3    1 12  7.75 2.416797      8     7.7 2.9652   4  12     8
## X14    4     e4    1 12  4.25 2.864358      4     4.1 2.9652   1   9     8
##           skew   kurtosis        se
## X11 -0.6227374 -0.5225309 0.5967081
## X12  0.1566396 -1.0817801 0.7017295
## X13  0.1261840 -1.3237827 0.6976693
## X14  0.4082316 -1.4971252 0.8268689
```

Des boites à moustache (avec les médianes et non les moyennes) sont toujours issues de la fonction générique `plot` :

```
plot (intra1m$erreurs ~ intra1m$essai)
```



11.4.3 Analyse de variance avec la fonction aov

L'analyse de variance pour mesures répétées ne peut pas se faire avec la fonction `lm`[13]. C'est donc la fonction `aov` du module de base qui permet de traiter ce type de données et il est indispensable de spécifier quel terme d'erreur utiliser pour tester l'effet des facteurs. Dans le cas d'une seule variable indépendante intraindividuelle comme ici, le terme d'erreur s'écrit `sujet/essai` (i.e. les essais sont "nichés" dans les sujets). Les commandes sont donc les suivantes :

```
summary (aov (erreurs ~ essai + Error (sujet/essai), data = intra1m))
```

```
##
## Error: sujet
##           Df Sum Sq Mean Sq F value Pr(>F)
## Residuals 11    181   16.45
##
## Error: sujet:essai
##           Df Sum Sq Mean Sq F value Pr(>F)
## essai      3  991.5   330.5   127.6 <2e-16 ***
## Residuals 33   85.5     2.6
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

[13]Comme un même individu est mesuré plusieurs fois, les observations ne sont plus indépendantes, ce qui viole un des postulats fondamentaux des modèles linéaires. Ceux-ci ont été élargis pour prendre en compte cette dépendance à travers les modèles linéaires à effets mixtes ou hiérarchiques, implémentées notamment dans le paquet `lme4` (Bates, Mächler, Bolker, & Walker (2015)). Ces modèles ne seront pas présentés dans ce chapitre.

Il est très important de ne pas oublier de spécifier le terme d'erreur . Sans celui-ci, c'est une analyse de variance inter-individuelle qui est réalisée (i.e. toutes les observations sont considérées comme relevant d'individus différents), comme le montre l'illustration suivante :

```
summary (aov (erreurs ~ essai, data = intra1m))
```

```
##             Df Sum Sq Mean Sq F value   Pr(>F)
## essai        3  991.5   330.5   54.57 7.15e-15 ***
## Residuals   44  266.5     6.1
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

On remarquera que, dans cette seconde analyse, n'apparaît pas la mention de la variance résiduelle due aux individus et que le nombre de degrés de liberté de la variance intra n'est pas la même :

- dans l'analyse intra-individuelle, ce nombre de degrés de liberté vaut (4 modalités - 1) * (12 individus - 1) = 33
- dans l'analyse inter-individuelle, ce nombre de degrés de liberté vaut (48 observations[14] - 4 modalités) = 44

11.4.4 Les comparaisons multiples

Les comparaisons multiples entre les moyennes dans le cas d'analyses de variances avec mesures répétées posent des questions très complexes du point de vue statistique et on trouve peu d'indications claires dans les manuels de méthodologie à ce sujet (beaucoup de textes sur l'analyse de variance pour mesures répétées ignorent même tout simplement la question). Les spécialistes ne sont pas d'accord entre eux et certains avouent même qu'ils n'ont pas de solution satisfaisante à proposer. L'un d'entre eux (D. Howell[15]), recommande de ne faire qu'un nombre réduit de comparaisons (celles qui sont justifiées d'un point de vue théorique) et de les ajuster avec la méthode de *Bonferroni*.

Si on applique cette procédure ici, et que l'on souhaite comparer le premier essai avec le second, le second avec le troisième et le troisième avec le quatrième, on fera trois tests de Student successifs (tests pour données appariées[16]). Pour conclure à l'existence d'une différence statistiquement significative au seuil de 5%, il faudra que la probabilité associée aux tests de Student soit inférieure à 0.05/3 soit 0.01667.

```
t.test (intra1m$erreurs[intra1m$essai == "e1"],
        intra1m$erreurs[intra1m$essai == "e2"], paired = T)
```

```
##
##  Paired t-test
##
## data:  intra1m$erreurs[intra1m$essai == "e1"] and intra1m$erreurs[intra1m$essai == "e2"]
## t = 7.543, df = 11, p-value = 1.138e-05
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  3.541037 6.458963
## sample estimates:
## mean of the differences
##                       5
```

```
t.test (intra1m$erreurs[intra1m$essai == "e2"],
        intra1m$erreurs[intra1m$essai == "e3"], paired = T)
```

```
##
```

[14] Dans l'analyse inter-individuelle, les 48 observations sont considérées comme 48 individus différents.
[15] Voir : http://www.uvm.edu/~dhowell/StatPages/More_Stuff/RepMeasMultComp/RepMeasMultComp.html
[16] Noter l'argument `paired = T`, dans la commande `t.test`

```
##  Paired t-test
##
## data:  intra1m$erreurs[intra1m$essai == "e2"] and intra1m$erreurs[intra1m$essai == "e3"]
## t = 8.7491, df = 11, p-value = 2.762e-06
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  2.806621 4.693379
## sample estimates:
## mean of the differences
##                    3.75
```

```
t.test (intra1m$erreurs[intra1m$essai == "e3"],
        intra1m$erreurs[intra1m$essai == "e4"], paired = T)
```

```
##
##  Paired t-test
##
## data:  intra1m$erreurs[intra1m$essai == "e3"] and intra1m$erreurs[intra1m$essai == "e4"]
## t = 6.7971, df = 11, p-value = 2.965e-05
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  2.36665 4.63335
## sample estimates:
## mean of the differences
##                     3.5
```

Réaliser ces tests de Student revient à ne prendre en compte que les données des deux essais comparés en faisant abstraction du reste des informations. Une autre solution consiste à effectuer les comparaisons en utilisant les informations (carré moyen et degré de liberté résiduels) de l'analyse de variance globale[17]. Cette procédure a été détaillée plus haut dans la section consacrée à la décomposition d'une interaction en effets simples. Appliquée aux trois comparaisons qui nous intéressent ici, on obtient (aux probabilités indiquées, on appliquera aussi l'ajustement de Bonferroni pour se prémunir des risques associés à la multiplicité des comparaisons) :

```
1 - pf ((((16.5 - 11.5)/sqrt (2.6))^2)*2,1,33)  # entre essai 1 et essai 2
```

```
## [1] 0.000111376
```

```
1 - pf ((((11.5 - 7.75)/sqrt (2.6))^2)*2,1,33)  # entre essai 2 et essai 3
```

```
## [1] 0.002395324
```

```
1 - pf ((((7.75 - 4.25)/sqrt (2.6))^2)*2,1,33)  # entre essai 3 et essai 4
```

```
## [1] 0.004265704
```

```
1 - pf ((((16.5 - 11.5)/sqrt (2.6))^2)*2,1,33)  # entre essai 1 et essai 2
```

```
## [1] 0.000111376
```

```
1 - pf ((((11.5 - 7.75)/sqrt (2.6))^2)*2,1,33)  # entre essai 2 et essai 3
```

```
## [1] 0.002395324
```

[17] Cette procédure n'est toutefois pas recommandée par Howell qui considère qu'elle est trop sensible aux violations des postulats de l'analyse de variance.

```
1 - pf ((((7.75 - 4.25)/sqrt (2.6))^2)*2,1,33)  # entre essai 3 et essai 4
```

```
## [1] 0.004265704
```

```
1 - pf ((((16.5 - 11.5)/sqrt (2.6))^2)*2,1,33)  # entre essai 1 et essai 2
```

```
## [1] 0.000111376
```

```
1 - pf ((((11.5 - 7.75)/sqrt (2.6))^2)*2,1,33)  # entre essai 2 et essai 3
```

```
## [1] 0.002395324
```

```
1 - pf ((((7.75 - 4.25)/sqrt (2.6))^2)*2,1,33)  # entre essai 3 et essai 4
```

```
## [1] 0.004265704
```

11.4.5 Le postulat de sphéricité dans l'analyse de variance avec mesures répétées

L'un des postulats les plus importants de l'analyse de variance avec mesures répétées est le postulat de "sphéricité". La sphéricité est une notion complexe qui fait référence au fait que les covariances des différences entre les mesures répétées doivent être égales. Le coefficient ϵ de *Greenhouse-Geisser* permet d'évaluer l'écart par rapport à la sphéricité. Il est égal à 1 lorsque la sphéricité est parfaite ; plus il est inférieur à 1, plus le postulat est compromis. La valeur minimale de ϵ dépend du nombre de modalités de la variable indépendante. Il serait ici de 1/(4-1), soit 0.333.

Le calcul de ce coefficient est un peu long[18]. Il faut d'abord extraire du tableau de données les vecteurs correspondant aux différentes modalités de la variable indépendante intraindividuelle à laquelle on s'intéresse :

```
e1 <- intra1m$erreurs[intra1m$essai == "e1"]
e2 <- intra1m$erreurs[intra1m$essai == "e2"]
e3 <- intra1m$erreurs[intra1m$essai == "e3"]
e4 <- intra1m$erreurs[intra1m$essai == "e4"]
```

Puis calculer la matrice de variance/covariance entre ces vecteurs :

```
S <- var (cbind (e1,e2,e3,e4))
S
```

```
##          e1       e2       e3       e4
## e1 4.272727 2.454545 1.227273 1.318182
## e2 2.454545 5.909091 4.772727 5.590909
## e3 1.227273 4.772727 5.840909 5.431818
## e4 1.318182 5.590909 5.431818 8.204545
```

Et enfin calculer la valeur de ϵ (dans la première ligne du script ci-dessous, il faut remplacer la valeur "4" par le nombre de modalités de la VI si celui-ci est différent) :

```
k <- 4
D <- k^2 * (mean (diag(S)) - mean (S)) ^2
N1 <- sum (S^2)
N2 <- 2 * k * sum (apply (S,1,mean)^2)
N3 <- k^2 * mean (S)^2
```

[18] Comme on le verra dans le prochain paragraphe, ce coefficient est calculé directement dans le paquet `ez`.

```
epsiGG <- D / ((k - 1) * (N1 - N2 + N3))
epsiGG
```

```
## [1] 0.6215803
```

La valeur trouvée pour ϵ est assez éloignée de 1. On peut craindre que le postulat de sphéricité soit violé. Il est alors possible de corriger la significativité du F de l'analyse de variance. Cela se fait de la manière suivante :

```
summary (aov (erreurs ~ essai + Error (sujet/essai), data = intra1m))
```

```
##
## Error: sujet
##           Df Sum Sq Mean Sq F value Pr(>F)
## Residuals 11    181   16.45
##
## Error: sujet:essai
##           Df Sum Sq Mean Sq F value Pr(>F)
## essai      3  991.5   330.5   127.6 <2e-16 ***
## Residuals 33   85.5     2.6
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
2 * (1 - pf(127.6, df1 = 3 * epsiGG, df2 = 33 * epsiGG))
```

```
## [1] 8.50009e-12
```

Dans la commande ci-dessus :

- la valeur "127.6" représente le F de la variable indépendante dans l'analyse de variance ;
- "df1" est le nombre de degrés de liberté du numérateur (i.e. celui de la variable indépendante) ;
- "df2" est le nombre de degrés de liberté du dénominateur (i.e. celui des résidus).

Le résultat indique que l'effet de l'entraînement reste très significatif, malgré la correction de *Greenhouse-Heiser*.

Dans la littérature spécialisée, on préfère parfois utiliser une autre correction, celle de *Huynh-Feldt*, qui est jugée moins conservatrice. On l'obtient de la manière suivante ("n" représente ci-dessous le nombre de sujets et "k" le nombre de modalités de la variable indépendante) :

```
n <- 12
k <- 4
epsiHF <- (n * (k-1) * epsiGG - 2) / ((k-1) * ((n-1) - (k-1) * epsiGG))
epsiHF
```

```
## [1] 0.7435253
```

La correction de la significativité du F par le coefficient de *Huynh-Feldt* est la suivante :

```
2 * (1 - pf(127.6, df1 = 3 * epsiHF, df2 = 33 * epsiHF))
```

```
## [1] 8.903989e-14
```

11.4.6 Analyse de variance avec le paquet `ez`

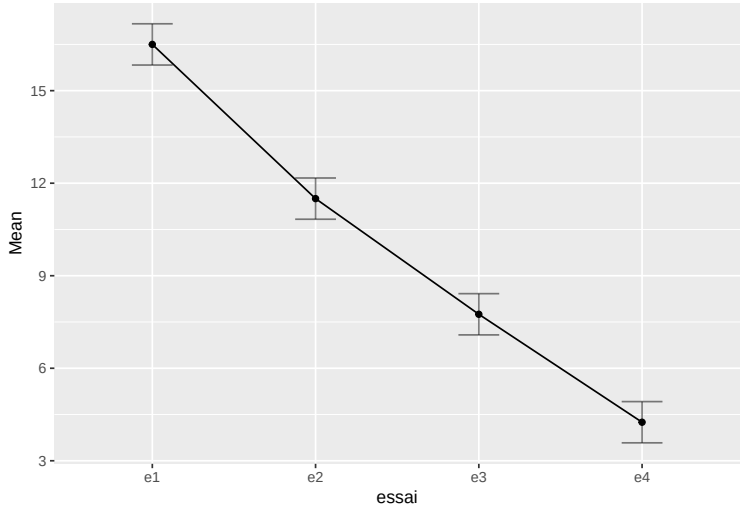
La fonction `ezANOVA` du paquet `ez` nécessite l'utilisation de l'argument `within` pour spécifier le nom de la variable indépendante intra-individuelle. Le reste de la commande est identique à ce que l'on a vu pour l'analyse de variables inter-individuelles. On notera que l'argument `detailed = T` fournit un test de sphéricité et les corrections de *Greenhouse-Geisser* ("GG") et de *Huynh-Feldt* ("HF").

```
ezANOVA(data = intra1m, wid = sujet, dv = erreurs, within = essai, detailed = T)
```

```
## $ANOVA
##        Effect DFn DFd    SSn   SSd        F            p p<.05       ges
## 1 (Intercept)   1  11 4800.0 181.0 291.7127 2.885118e-09     * 0.9473996
## 2       essai   3  33  991.5  85.5 127.5614 3.161251e-18     * 0.7881558
##
## $`Mauchly's Test for Sphericity`
##   Effect         W         p p<.05
## 2  essai 0.3980064 0.1124389
##
## $`Sphericity Corrections`
##   Effect       GGe        p[GG] p[GG]<.05       HFe        p[HF] p[HF]<.05
## 2  essai 0.6215803 4.262187e-12         * 0.7435253 4.469899e-14         *
```

La représentation graphique suit la même logique :

```
ezPlot(data = intra1m, wid = sujet, dv = erreurs, within = essai, x = essai, do_lines = T)
```



11.5 Analyse de variance avec deux VI intra-individuelles

Reprenons l'exemple précédent en imaginant que les sujets effectuent deux fois chaque jour la tâche pendant quatre jours consécutifs. Il y a donc deux variables indépendantes intraindividuelles : le jour (de 1 à 4) et le moment (le matin ou le soir).

11.5.1 L'organisation des données

Dans un format où chaque individu occupe une ligne, les données se présentent de la manière suivante :

```
intra2 <- data.frame (sujet = factor (1:12), J1m = c(18,19,14,16,12,18,16,18,16,19,16,16),
                      J1s = c(22,24,16,21,15,22,19,22,20,22,16,20),
                      J2m = c(14,12,10,12,8,10,10,8,12,16,14,12),
                      J2s = c(17,15,11,14,7,11,12,11,14,18,15,14),
                      J3m = c(12,8,6,10,6,5,8,4,6,10,10,8),
                      J3s = c(12,10,6,9,5,4,8,5,8,11,11,7),
                      J4m = c(6,4,2,4,2,1,4,1,2,8,9,8),
```

```
                    J4s = c(7,5,1,3,4,4,4,1,2,7,10,8))
intra2
```

```
##    sujet J1m J1s J2m J2s J3m J3s J4m J4s
## 1      1  18  22  14  17  12  12   6   7
## 2      2  19  24  12  15   8  10   4   5
## 3      3  14  16  10  11   6   6   2   1
## 4      4  16  21  12  14  10   9   4   3
## 5      5  12  15   8   7   6   5   2   4
## 6      6  18  22  10  11   5   4   1   4
## 7      7  16  19  10  12   8   8   4   4
## 8      8  18  22   8  11   4   5   1   1
## 9      9  16  20  12  14   6   8   2   2
## 10    10  19  22  16  18  10  11   8   7
## 11    11  16  16  14  15  10  11   9  10
## 12    12  16  20  12  14   8   7   8   8
```

Ce format (une ligne par individu) est inadéquat pour l'analyse avec R. Il faut le transformer (fonction `melt` du paquet `reshape2`) de manière à ce que chaque ligne corresponde à une seule observation :

```
intra2m <- melt (intra2, id.vars = "sujet", variable.name = "essai",
                 value.name = "erreurs")
```

Le nouveau tableau de données ("intra2m") dont on ne montre que les 15 premières lignes ci-dessous comporte 96 lignes (12 individus $\times$ 4 jours $\times$ 2 moments) :

```
head (intra2m, 15)
```

```
##    sujet essai erreurs
## 1      1   J1m      18
## 2      2   J1m      19
## 3      3   J1m      14
## 4      4   J1m      16
## 5      5   J1m      12
## 6      6   J1m      18
## 7      7   J1m      16
## 8      8   J1m      18
## 9      9   J1m      16
## 10    10   J1m      19
## 11    11   J1m      16
## 12    12   J1m      16
## 13     1   J1s      22
## 14     2   J1s      24
## 15     3   J1s      16
```

Dans ce tableau, il n'y a qu'une seule variable indépendante intra-individuelle ("essai") qui comporte 8 modalités ("J1m,"J1s", ... "J4s"). Ce que l'on souhaite, c'est avoir deux variables qui repèrent pour l'une le jour et l'autre le moment. La méthode la plus rapide consiste à utiliser la fonction `substr` qui extrait d'une chaine de caractères les éléments en fonction de leur position. En écrivant, par exemple `substr (x, start = 1, stop = 2)`, on extrait de l'objet `x` les deux premiers caractères. On peut alors facilement créer les deux variables nécessaires :

```
intra2m$jour <- factor (substr(intra2m$essai, start = 1, stop = 2))
intra2m$moment <- factor (substr(intra2m$essai, start = 3, stop = 3))
head (intra2m, 15)  # affichage des 15 premières lignes
```

```
##    sujet essai erreurs jour moment
```

```
## 1      1   J1m      18   J1      m
## 2      2   J1m      19   J1      m
## 3      3   J1m      14   J1      m
## 4      4   J1m      16   J1      m
## 5      5   J1m      12   J1      m
## 6      6   J1m      18   J1      m
## 7      7   J1m      16   J1      m
## 8      8   J1m      18   J1      m
## 9      9   J1m      16   J1      m
## 10    10   J1m      19   J1      m
## 11    11   J1m      16   J1      m
## 12    12   J1m      16   J1      m
## 13     1   J1s      22   J1      s
## 14     2   J1s      24   J1      s
## 15     3   J1s      16   J1      s
```

Dans l'affichage ci-dessus, seules les 15 premières lignes du tableau de données apparaissent, mais le nouveau tableau comporte bien 96 lignes :

```
str (intra2m)
```

```
## 'data.frame':    96 obs. of  5 variables:
##  $ sujet  : Factor w/ 12 levels "1","2","3","4",..: 1 2 3 4 5 6 7 8 9 10 ...
##  $ essai  : Factor w/ 8 levels "J1m","J1s","J2m",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ erreurs: num  18 19 14 16 12 18 16 18 16 19 ...
##  $ jour   : Factor w/ 4 levels "J1","J2","J3",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ moment : Factor w/ 2 levels "m","s": 1 1 1 1 1 1 1 1 1 1 ...
```

11.5.2 Statistiques descriptives

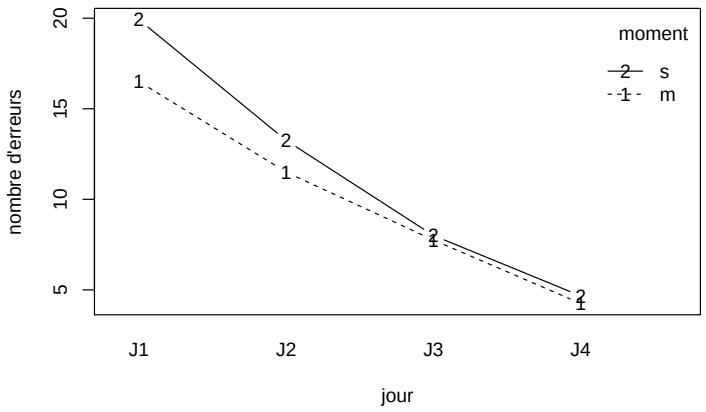
La fonction `describeBy` du paquet `psych` fournit les statistiques univariées correspondant aux 8 moments :

```
describeBy (intra2m$erreurs, list (intra2m$jour, intra2m$moment), mat = T)
```

```
##     item group1 group2 vars  n      mean       sd median trimmed    mad
## X11    1     J1      m    1 12 16.500000 2.067058   16.0    16.7 2.9652
## X12    2     J2      m    1 12 11.500000 2.430862   12.0    11.4 2.9652
## X13    3     J3      m    1 12  7.750000 2.416797    8.0     7.7 2.9652
## X14    4     J4      m    1 12  4.250000 2.864358    4.0     4.1 2.9652
## X15    5     J1      s    1 12 19.916667 2.874918   20.5    20.0 2.2239
## X16    6     J2      s    1 12 13.250000 3.018880   14.0    13.4 3.7065
## X17    7     J3      s    1 12  8.000000 2.662876    8.0     8.0 3.7065
## X18    8     J4      s    1 12  4.666667 2.839121    4.0     4.5 3.7065
##     min max range       skew   kurtosis        se
## X11  12  19     7 -0.6227374 -0.5225309 0.5967081
## X12   8  16     8  0.1566396 -1.0817801 0.7017295
## X13   4  12     8  0.1261840 -1.3237827 0.6976693
## X14   1   9     8  0.4082316 -1.4971252 0.8268689
## X15  15  24     9 -0.4708708 -1.2794122 0.8299172
## X16   7  18    11 -0.3282542 -0.6605464 0.8714756
## X17   4  12     8  0.0000000 -1.5581032 0.7687061
## X18   1  10     9  0.3317711 -1.1887166 0.8195835
```

Les moyennes peuvent donner lieu à une représentation graphique permettant de repérer visuellement la présence d'une éventuelle interaction :

```
interaction.plot (intra2m$jour, intra2m$moment, intra2m$erreurs, type = "b",
                  xlab = "jour", trace.label = "moment", ylab = "nombre d'erreurs")
```



11.5.3 Analyse de variance avec la fonction aov

L'analyse de variance s'obtient en utilisant la même logique que dans le cas d'une seule variable indépendante intraindividuelle[19] :

```
summary (aov (erreurs ~ (jour * moment) + Error (sujet/(jour * moment)), data = intra2m))
```

```
##
## Error: sujet
##           Df Sum Sq Mean Sq F value Pr(>F)
## Residuals 11    385      35
##
## Error: sujet:jour
##           Df Sum Sq Mean Sq F value Pr(>F)
## jour       3 2546.8   848.9   135.9 <2e-16 ***
## Residuals 33  206.2     6.2
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Error: sujet:moment
##           Df Sum Sq Mean Sq F value   Pr(>F)
## moment     1  51.04   51.04   59.36 9.32e-06 ***
## Residuals 11   9.46    0.86
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Error: sujet:jour:moment
##             Df Sum Sq Mean Sq F value   Pr(>F)
## jour:moment  3  38.79  12.931      18 4.25e-07 ***
## Residuals   33  23.71   0.718
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

[19] Noter le terme d'erreur utilisé : `Error (sujet/(jour * moment))`.

11.5.4 Comparaisons multiples

L'analyse de variance montre que les trois effets sont statistiquement significatifs : le jour, le moment et l'interaction entre les deux. D'une part, la performance est meilleure le matin que le soir (le F de la variable "moment" est statistiquement significatif) et d'autre part, les résultats s'améliorent régulièrement entre le jour 1 et le jour 4 (voir le F de la variable "jour") comme le montrent les résultats suivants :

```
describeBy (intra2m$erreurs, intra2m$moment, mat = T)
```

```
##     item group1 vars  n     mean       sd median trimmed    mad min max
## X11    1      m    1 48 10.00000 5.173583     10  10.000 5.9304   1  19
## X12    2      s    1 48 11.45833 6.444388     11  11.325 7.4130   1  24
##     range       skew  kurtosis        se
## X11    18 0.03520468 -1.068919 0.7467423
## X12    23 0.24547401 -1.090085 0.9301673
```

```
describeBy (intra2m$erreurs, intra2m$jour, mat = T)
```

```
##     item group1 vars  n      mean       sd median trimmed    mad min max
## X11    1     J1    1 24 18.208333 3.006936     18   18.25 2.9652  12  24
## X12    2     J2    1 24 12.375000 2.825543     12   12.35 2.9652   7  18
## X13    3     J3    1 24  7.875000 2.490198      8    7.85 2.9652   4  12
## X14    4     J4    1 24  4.458333 2.797191      4    4.30 2.9652   1  10
##     range       skew   kurtosis        se
## X11    12 0.03802120 -0.9072184 0.6137883
## X12    11 0.02060635 -0.7546217 0.5767616
## X13     8 0.07158874 -1.3108500 0.5083096
## X14     9 0.39074946 -1.1944462 0.5709741
```

En ce qui concerne la variable "jour", on peut procéder à des comparaisons de moyennes avec un test de Student pour données appariées :

```
t.test (intra2m$erreurs[intra2m$jour == "J1"], intra2m$erreurs[intra2m$jour == "J2"], paired = T)
```

```
##
##  Paired t-test
##
## data:  intra2m$erreurs[intra2m$jour == "J1"] and intra2m$erreurs[intra2m$jour == "J2"]
## t = 10.659, df = 23, p-value = 2.264e-10
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  4.701196 6.965471
## sample estimates:
## mean of the differences
##                5.833333
```

```
t.test (intra2m$erreurs[intra2m$jour == "J2"], intra2m$erreurs[intra2m$jour == "J3"], paired = T)
```

```
##
##  Paired t-test
##
## data:  intra2m$erreurs[intra2m$jour == "J2"] and intra2m$erreurs[intra2m$jour == "J3"]
## t = 13.427, df = 23, p-value = 2.274e-12
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  3.80671 5.19329
```

```
## sample estimates:
## mean of the differences
##                     4.5
```

```
t.test (intra2m$erreurs[intra2m$jour == "J3"], intra2m$erreurs[intra2m$jour == "J4"], paired = T)
```

```
##
##  Paired t-test
##
## data:  intra2m$erreurs[intra2m$jour == "J3"] and intra2m$erreurs[intra2m$jour == "J4"]
## t = 8.034, df = 23, p-value = 3.987e-08
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  2.536922 4.296412
## sample estimates:
## mean of the differences
##                3.416667
```

En ce qui concerne l'interaction, il est intéressant ici d'examiner si les différences entre le moment de la journée se reproduisent de la même manière tous les jours. A l'évidence, ce n'est pas le cas. La différence de performances entre matin et soir ne s'observe (de manière significative) que les jours 1 et 2. A partir du 3e jour, il n'y a plus de différences significatives entre les deux moments comment le montrent les résultats ci-dessous :

```
t.test (intra2m$erreurs[intra2m$jour == "J1" & intra2m$moment == "m"],
        intra2m$erreurs[intra2m$jour == "J1" & intra2m$moment == "s"], paired = T)
```

```
##
##  Paired t-test
##
## data:  intra2m$erreurs[intra2m$jour == "J1" & intra2m$moment == "m"] and intra2m$erreurs[intra2m$jou
## t = -8.5831, df = 11, p-value = 3.326e-06
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -4.292812 -2.540521
## sample estimates:
## mean of the differences
##               -3.416667
```

```
t.test (intra2m$erreurs[intra2m$jour == "J2" & intra2m$moment == "m"],
        intra2m$erreurs[intra2m$jour == "J2" & intra2m$moment == "s"], paired = T)
```

```
##
##  Paired t-test
##
## data:  intra2m$erreurs[intra2m$jour == "J2" & intra2m$moment == "m"] and intra2m$erreurs[intra2m$jou
## t = -5.3262, df = 11, p-value = 0.0002425
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -2.473165 -1.026835
## sample estimates:
## mean of the differences
##                   -1.75
```

```
t.test (intra2m$erreurs[intra2m$jour == "J3" & intra2m$moment == "m"],
        intra2m$erreurs[intra2m$jour == "J3" & intra2m$moment == "s"], paired = T)
```

```
##
##  Paired t-test
##
## data:  intra2m$erreurs[intra2m$jour == "J3" & intra2m$moment == "m"] and intra2m$erreurs[intra2m$jou
## t = -0.76089, df = 11, p-value = 0.4627
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -0.9731653  0.4731653
## sample estimates:
## mean of the differences
##                   -0.25
```

```
t.test (intra2m$erreurs[intra2m$jour == "J4" & intra2m$moment == "m"],
        intra2m$erreurs[intra2m$jour == "J4" & intra2m$moment == "s"], paired = T)
```

```
##
##  Paired t-test
##
## data:  intra2m$erreurs[intra2m$jour == "J4" & intra2m$moment == "m"] and intra2m$erreurs[intra2m$jou
## t = -1.1639, df = 11, p-value = 0.2691
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -1.2045965  0.3712632
## sample estimates:
## mean of the differences
##              -0.4166667
```

11.5.5 Le postulat de sphéricité

L'ajustement du résultat de l'analyse de variance pour tenir compte du non-respect du postulat de sphéricité peut se faire de la même manière que dans le cas d'une seule variable indépendante intraindividuelle. Cet ajustement n'est pas nécessaire pour la VI "moment" car elle ne comprend que deux modalités[20]. Le script ci-dessous présente l'ajustement de *Greenhouse-Geisser* pour le test de l'effet de la variable indépendante "jour"[21] :

```
tapply (intra2m$erreurs, list (intra2m$sujet, intra2m$jour), mean)
```

```
##      J1   J2   J3  J4
## 1  20.0 15.5 12.0 6.5
## 2  21.5 13.5  9.0 4.5
## 3  15.0 10.5  6.0 1.5
## 4  18.5 13.0  9.5 3.5
## 5  13.5  7.5  5.5 3.0
## 6  20.0 10.5  4.5 2.5
## 7  17.5 11.0  8.0 4.0
## 8  20.0  9.5  4.5 1.0
## 9  18.0 13.0  7.0 2.0
## 10 20.5 17.0 10.5 7.5
## 11 16.0 14.5 10.5 9.5
## 12 18.0 13.0  7.5 8.0
```

```
# extraction de la moyenne des erreurs à J1 pour chaque individu
e1 <- tapply (intra2m$erreurs, list (intra2m$sujet, intra2m$jour), mean)[,1]
```

[20] La question de la sphéricité ne se pose qu'à partir de trois modalités.

[21] A nouveau, ces corrections sont fournies automatiquement avec le paquet `ez`, ce qui évite ces calculs fastidieux.

```
# extraction de la moyenne des erreurs à J2 pour chaque individu
e2 <- tapply (intra2m$erreurs, list (intra2m$sujet, intra2m$jour), mean)[,2]
# extraction de la moyenne des erreurs à J3 pour chaque individu
e3 <- tapply (intra2m$erreurs, list (intra2m$sujet, intra2m$jour), mean)[,3]
# extraction de la moyenne des erreurs à J4 pour chaque individu
e4 <- tapply (intra2m$erreurs, list (intra2m$sujet, intra2m$jour), mean)[,4]
S <- var (cbind (e1,e2,e3,e4))  # calcul de la matrice de covariance entre les jours
k <- 4  # nombre de modalités de la variable jour
D <- k^2 * (mean (diag(S)) - mean (S)) ^2
N1 <- sum (S^2)
N2 <- 2 * k * sum (apply (S,1,mean)^2)
N3 <- k^2 * mean (S)^2
epsiGG <- D / ((k - 1) * (N1 - N2 + N3))  # calcul du coefficient de Greenhouse-Geisser
epsiGG
```

```
## [1] 0.5962762
```

```
# valeur corrigée de la significativité du F
2 * (1 - pf(135.86, df1 = 3 * epsiGG, df2 = 33 * epsiGG))
```

```
## [1] 1.239631e-11
```

11.5.6 Analyse de variance avec le paquet ez

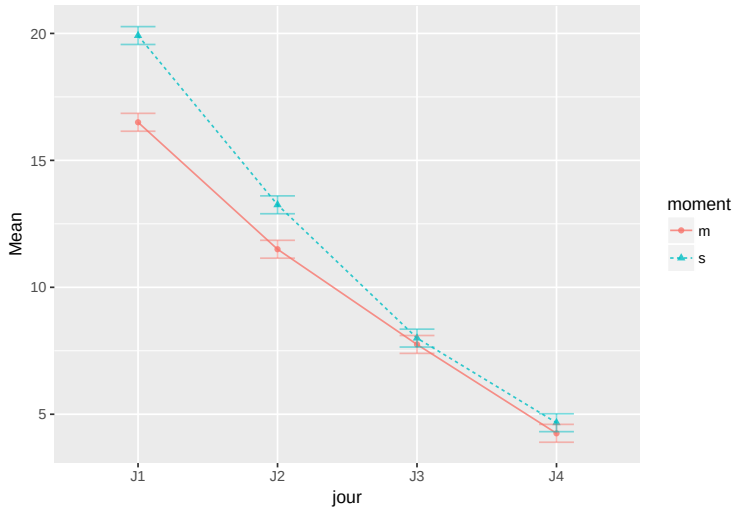
La fonction `ezANOVA` du paquet `ez` fournit directement le tableau d'analyse de variance, ainsi que les coefficients de correction en cas de violation de la sphéricité :

```
ezANOVA(data = intra2m, wid = sujet, dv = erreurs, within = .(jour, moment), detailed = T)
```

```
## $ANOVA
##        Effect DFn DFd         SSn        SSd         F            p p<.05
## 1 (Intercept)   1  11 11051.04167 384.958333 315.77822 1.892051e-09     *
## 2        jour   3  33  2546.79167 206.208333 135.85633 1.213996e-18     *
## 3      moment   1  11    51.04167   9.458333  59.36123 9.324244e-06     *
## 4 jour:moment   3  33    38.79167  23.708333  17.99824 4.251564e-07     *
##          ges
## 1 0.94652563
## 2 0.80311929
## 3 0.07557530
## 4 0.05849827
##
## $`Mauchly's Test for Sphericity`
##        Effect         W         p p<.05
## 2        jour 0.3279266 0.0559174
## 4 jour:moment 0.4685987 0.1967868
##
## $`Sphericity Corrections`
##        Effect       GGe        p[GG] p[GG]<.05       HFe        p[HF]
## 2        jour 0.5962762 6.199642e-12         * 0.7044325 9.793483e-14
## 4 jour:moment 0.7686579 6.827440e-06         * 0.9842652 5.131851e-07
##   p[HF]<.05
## 2         *
## 4         *
```

La représentation graphique :

```
ezPlot(data = intra2m, wid = sujet, dv = erreurs, within = .(jour,moment), x = jour,
       split = moment, do_lines = T)
```



11.6 Analyse de variance avec deux VI intra et une VI inter

Dans l'étude servant d'exemple ici, on s'intéresse à des personnes (n = 14) venant de perdre leur conjoint, à qui on propose des séances de psychothérapie pour accompagner leur deuil. Ces séances ont lieu à quatre reprises sur une durée totale de deux semaines. L'humeur négative des participants est mesurée en début et en fin de chaque séance. Les deux variables intraindividuelles sont la séance (de 1 à 4) et le moment de la mesure (pré ou post). La variable interindividuelle est le sexe et la variable dépendante est l'humeur négative ("HN").

11.6.1 L'organisation des données

Dans un format où chaque individu occupe une ligne, les données se présentent de la manière suivante :

```
mixte <- data.frame (sujet = 1:14, sexe = c(1,2,1,2,1,2,1,2,1,2,1,2,1,2),
                     s1pre = c(15,14,24,26,29,38,24,14,20,18,17,24,28,35),
                     s1post = c(16,14,24,30,34,39,20,17,22,21,19,30,28,40),
                     s2pre = c(18,14,22,33,23,31,15,14,23,17,15,24,28,20),
                     s2post = c(21,14,25,34,48,29,15,14,22,16,19,25,34,21),
                     s3pre = c(17,14,18,29,23,39,15,14,16,15,14,20,34,22),
                     s3post = c(21,14,26,34,46,34,14,14,19,14,16,25,36,36),
                     s4pre = c(18,14,23,24,32,37,14,14,21,14,14,18,34,20),
                     s4post = c(17,14,33,25,50,34,44,14,26,14,15,20,33,31))
mixte
```

```
##    sujet sexe s1pre s1post s2pre s2post s3pre s3post s4pre s4post
## 1      1    1    15     16    18     21    17     21    18     17
## 2      2    2    14     14    14     14    14     14    14     14
## 3      3    1    24     24    22     25    18     26    23     33
## 4      4    2    26     30    33     34    29     34    24     25
## 5      5    1    29     34    23     48    23     46    32     50
## 6      6    2    38     39    31     29    39     34    37     34
## 7      7    1    24     20    15     15    15     14    14     44
```

```
## 8      8    2    14     17     14     14     14     14     14     14
## 9      9    1    20     22     23     22     16     19     21     26
## 10    10    2    18     21     17     16     15     14     14     14
## 11    11    1    17     19     15     19     14     16     14     15
## 12    12    2    24     30     24     25     20     25     18     20
## 13    13    1    28     28     28     34     34     36     34     33
## 14    14    2    35     40     20     21     22     36     20     31
```

Ce format (une ligne par individu) est inadéquat pour l'analyse avec R. Il faut le transformer (fonction `melt` du paquet `reshape2`) de manière à ce que chaque ligne corresponde à une seule observation[22] :

```
mixtem <- melt (mixte, id.vars = c("sujet", "sexe"), variable.name = "mesure",
                value.name = "HN")
```

Le nouveau tableau de données ("mixtem") dont on ne montre que les 20 premières lignes ci-dessous comporte 112 lignes (14 individus $\times$ 4 séances $\times$ 2 moments) :

```
head (mixtem, 20)
```

```
##    sujet sexe mesure HN
## 1      1    1  s1pre 15
## 2      2    2  s1pre 14
## 3      3    1  s1pre 24
## 4      4    2  s1pre 26
## 5      5    1  s1pre 29
## 6      6    2  s1pre 38
## 7      7    1  s1pre 24
## 8      8    2  s1pre 14
## 9      9    1  s1pre 20
## 10    10    2  s1pre 18
## 11    11    1  s1pre 17
## 12    12    2  s1pre 24
## 13    13    1  s1pre 28
## 14    14    2  s1pre 35
## 15     1    1 s1post 16
## 16     2    2 s1post 14
## 17     3    1 s1post 24
## 18     4    2 s1post 30
## 19     5    1 s1post 34
## 20     6    2 s1post 39
```

Le nouveau tableau ne comporte qu'une seule variable indépendante intra-individuelle qui combine la séance et le moment. Le script suivant permet de créer deux variables séparées : la séance et le moment :

```
mixtem$seance <- factor (substr(mixtem$mesure, start = 1, stop = 2))
mixtem$moment <- factor (substr(mixtem$mesure, start = 3, stop = 6))
head (mixtem, 20)  # Les 20 premières lignes du tableau de données
```

```
##    sujet sexe mesure HN seance moment
## 1      1    1  s1pre 15     s1    pre
## 2      2    2  s1pre 14     s1    pre
## 3      3    1  s1pre 24     s1    pre
## 4      4    2  s1pre 26     s1    pre
## 5      5    1  s1pre 29     s1    pre
## 6      6    2  s1pre 38     s1    pre
```

[22] On notera que le "sexe" qui est la variable inter-individuelle figure dans la liste des variables d'identifiants (argument `id.vars`).

```
## 7      7    1  s1pre 24     s1    pre
## 8      8    2  s1pre 14     s1    pre
## 9      9    1  s1pre 20     s1    pre
## 10    10    2  s1pre 18     s1    pre
## 11    11    1  s1pre 17     s1    pre
## 12    12    2  s1pre 24     s1    pre
## 13    13    1  s1pre 28     s1    pre
## 14    14    2  s1pre 35     s1    pre
## 15     1    1 s1post 16     s1   post
## 16     2    2 s1post 14     s1   post
## 17     3    1 s1post 24     s1   post
## 18     4    2 s1post 30     s1   post
## 19     5    1 s1post 34     s1   post
## 20     6    2 s1post 39     s1   post
```

11.6.2 Statistiques descriptives

Il y a dans cet exemple 16 moyennes différentes (les huit mesures d'humeur négative pour les hommes et pour les femmes) :

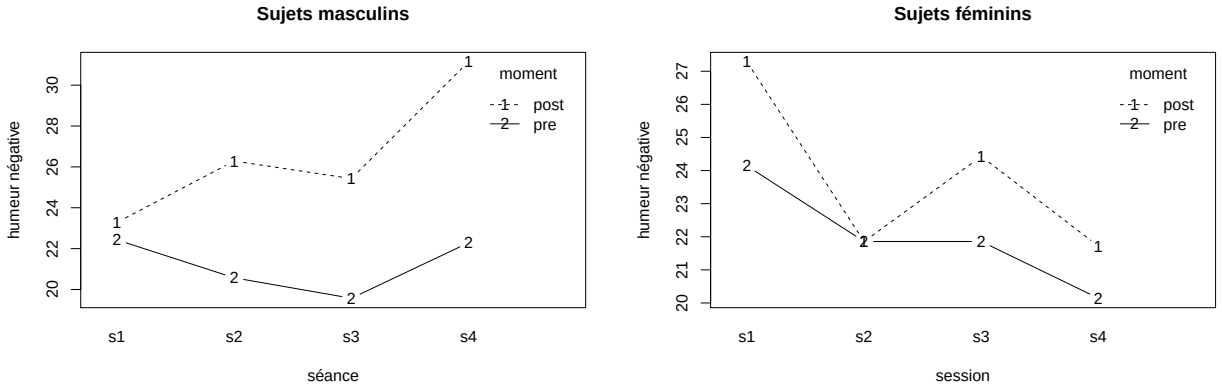
```
describeBy(mixtem$HN, list (mixtem$sexe, mixtem$seance, mixtem$moment), mat = T)
```

```
##      item group1 group2 group3 vars n     mean        sd median  trimmed
## X11     1      1     s1   post    1 7 23.28571  6.074929     22 23.28571
## X12     2      2     s1   post    1 7 27.28571 10.291004     30 27.28571
## X13     3      1     s2   post    1 7 26.28571 11.250397     22 26.28571
## X14     4      2     s2   post    1 7 21.85714  7.819390     21 21.85714
## X15     5      1     s3   post    1 7 25.42857 11.659862     21 25.42857
## X16     6      2     s3   post    1 7 24.42857 10.357882     25 24.42857
## X17     7      1     s4   post    1 7 31.14286 13.005493     33 31.14286
## X18     8      2     s4   post    1 7 21.71429  8.459990     20 21.71429
## X19     9      1     s1    pre    1 7 22.42857  5.318432     24 22.42857
## X110   10      2     s1    pre    1 7 24.14286  9.633770     24 24.14286
## X111   11      1     s2    pre    1 7 20.57143  4.790864     22 20.57143
## X112   12      2     s2    pre    1 7 21.85714  7.776644     20 21.85714
## X113   13      1     s3    pre    1 7 19.57143  6.996598     17 19.57143
## X114   14      2     s3    pre    1 7 21.85714  9.299258     20 21.85714
## X115   15      1     s4    pre    1 7 22.28571  8.056349     21 22.28571
## X116   16      2     s4    pre    1 7 20.14286  8.335238     18 20.14286
##          mad min max range         skew   kurtosis       se
## X11   4.4478  16  34    18  0.530179559 -1.2224592 2.296107
## X12  13.3434  14  40    26  0.002776707 -1.8666452 3.889634
## X13   4.4478  15  48    33  0.867277957 -0.8061433 4.252250
## X14  10.3782  14  34    20  0.303058311 -1.7297389 2.955452
## X15   7.4130  14  46    32  0.649195875 -1.3101996 4.407014
## X16  16.3086  14  36    22 -0.012750561 -2.1287573 3.914911
## X17  16.3086  15  50    35  0.108584626 -1.6896807 4.915614
## X18   8.8956  14  34    20  0.308660802 -1.8575053 3.197576
## X19   5.9304  15  29    14 -0.113722202 -1.7877976 2.010178
## X110 14.8260  14  38    24  0.260168458 -1.7831490 3.641223
## X111  5.9304  15  28    13  0.102129344 -1.6197942 1.810777
## X112  8.8956  14  33    19  0.323965412 -1.8041362 2.939295
## X113  2.9652  14  34    20  1.130003217 -0.3118842 2.644465
## X114  8.8956  14  39    25  0.728953279 -1.1123578 3.514789
```

```
## X115 10.3782  14  34    20  0.356815628 -1.7282254 3.045014
## X116  5.9304  14  37    23  1.021862107 -0.4765767 3.150424
```

Comme il y a trois variables indépendantes, il est plus lisible de faire deux graphiques différents : un pour les hommes et un autre pour les femmes. Cela peut s'obtenir de la manière suivante :

```
interaction.plot (mixtem$seance[mixtem$sexe == 1], mixtem$moment[mixtem$sexe == 1],
                  mixtem$HN[mixtem$sexe == 1], xlab = "séance", ylab = "humeur négative",
                  trace.label = "moment", type = "b", main = "Sujets masculins")
interaction.plot (mixtem$seance[mixte$sexe == 2], mixtem$moment[mixte$sexe == 2],
                  mixtem$HN[mixte$sexe == 2], xlab = "session", ylab = "humeur négative",
                  trace.label = "moment", type = "b", main = "Sujets féminins")
```



11.6.3 Analyse de variance avec la fonction aov

Avant de procéder à l'analye de variance, il est toujours souhaitable de vérifier que les différentes variables sont définies correctement. Cette vérification s'avère particulièrement utile ici, car on s'aperçoit que ce n'est pas le cas :

```
str (mixtem)
```

```
## 'data.frame':    112 obs. of  6 variables:
##  $ sujet : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ sexe  : num  1 2 1 2 1 2 1 2 1 2 ...
##  $ mesure: Factor w/ 8 levels "s1pre","s1post",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ HN    : num  15 14 24 26 29 38 24 14 20 18 ...
##  $ seance: Factor w/ 4 levels "s1","s2","s3",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ moment: Factor w/ 2 levels "post","pre": 2 2 2 2 2 2 2 2 2 2 ...
```

Les variables "sujet" et "sexe" doivent être déclarées comme des facteurs :

```
mixtem$sujet <- factor (mixtem$sujet)
mixtem$sexe <- factor (mixtem$sexe)
```

Avec la fonction `aov`, la commande à utiliser est la suivante. On notera que le terme d'erreur spécifié est : `Error (sujet/(seance * moment))` :

```
summary (aov (HN ~ (sexe * seance * moment) + Error (sujet/(seance*moment)), data = mixtem))
```

```
##
## Error: sujet
##           Df Sum Sq Mean Sq F value Pr(>F)
## sexe       1     26    26.0   0.053  0.822
```

```
## Residuals 12   5893   491.1
##
## Error: sujet:seance
##             Df Sum Sq Mean Sq F value Pr(>F)
## seance       3   52.4   17.45   0.785  0.510
## sexe:seance  3  285.6   95.20   4.284  0.011 *
## Residuals   36  800.0   22.22
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Error: sujet:moment
##             Df Sum Sq Mean Sq F value Pr(>F)
## moment       1  357.1   357.1   8.177 0.0144 *
## sexe:moment  1   85.7    85.7   1.963 0.1865
## Residuals   12  524.1    43.7
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Error: sujet:seance:moment
##                    Df Sum Sq Mean Sq F value Pr(>F)
## seance:moment       3   42.6   14.21   0.992  0.408
## sexe:seance:moment  3   92.3   30.77   2.147  0.111
## Residuals          36  516.0   14.33
```

Les résultats concernant les différents effets sont résumés ci-dessous :

- sexe : F(1,12) = 0.053, p = 0.822
- **moment : F(1,12) = 8.177, p = 0.014**
- moment:sexe : F(1,12) = 1.963, p = 0.141
- seance : F(3,36) = 0.785, p = 0.510
- **seance:sexe : F(3,36) = 4.284, p = 0.011**
- seance:moment : F(3,36) = 0.992, p = 0.408
- seance:moment:sexe : F(3,36) = 2.147, p = 0.111

11.6.4 Analyse de variance avec le paquet ez

Nous avons vu dans le paragraphe précédent qu'il fallait vérifier (et éventuellement le faire si ce n'est pas le cas) si les variables indépendantes et la variable identifiant les sujets sont bien définies comme des "facteurs", ce qui est le cas maintenant :

```
str (mixtem)
```

```
## 'data.frame':    112 obs. of  6 variables:
##  $ sujet : Factor w/ 14 levels "1","2","3","4",..: 1 2 3 4 5 6 7 8 9 10 ...
##  $ sexe  : Factor w/ 2 levels "1","2": 1 2 1 2 1 2 1 2 1 2 ...
##  $ mesure: Factor w/ 8 levels "s1pre","s1post",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ HN    : num  15 14 24 26 29 38 24 14 20 18 ...
##  $ seance: Factor w/ 4 levels "s1","s2","s3",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ moment: Factor w/ 2 levels "post","pre": 2 2 2 2 2 2 2 2 2 2 ...
```

L'utilisation de la fonction `ezANOVA` ne présente pas de difficultés particulières :

```
ezANOVA(data = mixtem, wid = sujet, dv = HN, between = sexe,
        within = .(seance, moment),detailed = T)
```

```
## $ANOVA
```

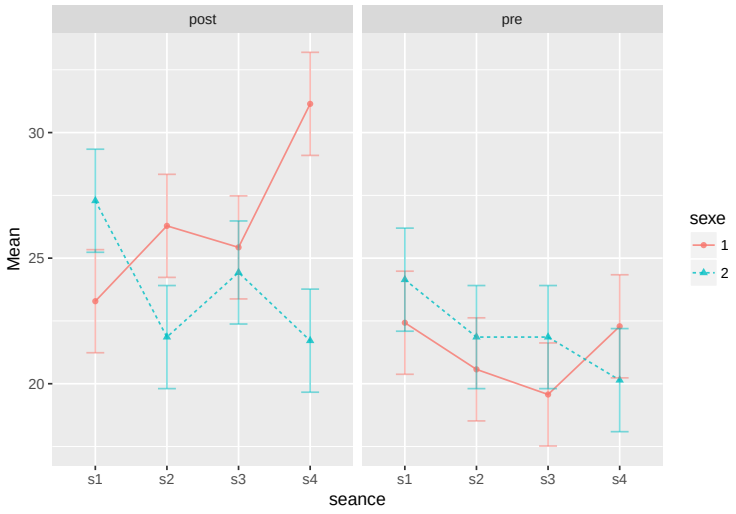
```
##               Effect DFn DFd         SSn       SSd            F
## 1        (Intercept)   1  12 61289.28571 5892.6786 124.81105488
## 2               sexe   1  12    26.03571 5892.6786   0.05301979
## 3             seance   3  36    52.35714  800.0357   0.78532208
## 5             moment   1  12   357.14286  524.1071   8.17717206
## 4        sexe:seance   3  36   285.60714  800.0357   4.28391590
## 6        sexe:moment   1  12    85.75000  524.1071   1.96333901
## 7      seance:moment   3  36    42.64286  516.0357   0.99162572
## 8 sexe:seance:moment   3  36    92.32143  516.0357   2.14686137
##              p p<.05         ges
## 1 1.068234e-07     * 0.887965560
## 2 8.217681e-01       0.003355597
## 3 5.099877e-01       0.006725203
## 5 1.436515e-02     * 0.044146212
## 4 1.100878e-02     * 0.035618684
## 6 1.864849e-01       0.010967427
## 7 4.077283e-01       0.005484259
## 8 1.113513e-01       0.011797996
##
## $`Mauchly's Test for Sphericity`
##               Effect         W          p p<.05
## 3             seance 0.6401284 0.44477121
## 4        sexe:seance 0.6401284 0.44477121
## 7      seance:moment 0.3873699 0.07179212
## 8 sexe:seance:moment 0.3873699 0.07179212
##
## $`Sphericity Corrections`
##               Effect       GGe      p[GG] p[GG]<.05       HFe     p[HF]
## 3             seance 0.7701647 0.48254680           0.9644945 0.5061876
## 4        sexe:seance 0.7701647 0.01968183         * 0.9644945 0.0120353
## 7      seance:moment 0.6174752 0.38106615           0.7253469 0.3905699
## 8 sexe:seance:moment 0.6174752 0.14322670           0.7253469 0.1335452
##   p[HF]<.05
## 3
## 4         *
## 7
## 8
```

La représentation graphique se fait avec la fonction `ezPlot` dans laquelle on introduit un nouvel argument (`col`) qui permet de montrer des graphiques côte à côte en fonction des modalités d'une variable. Ici c'est la variable "moment" qui est utilisée dans ce cadre[23] :

```
ezPlot(data = mixtem, wid = sujet, dv = HN, between = sexe, within = .(seance, moment),
       x = seance, split = sexe, col = moment, do_lines = T)
```

```
## Warning: Mixed within-and-between-Ss effect requested; FLSD is only
## appropriate for within-Ss comparisons (see warning in ?ezStats or ?ezPlot).
```

[23] Il existe aussi l'argument `row` qui permet de positionner les graphiques les uns en dessous des autres en fonction des modalités d'une variable.



11.6.5 Synthèse des résultats

Sur les sept effets testés, deux sont donc seulement statistiquement significatifs :

- le moment de la séance
- l'interaction entre le sexe et la séance

En ce qui concerne le moment de la séance, l'effet est facile à décrire puisqu'il n'y a que deux moyennes :

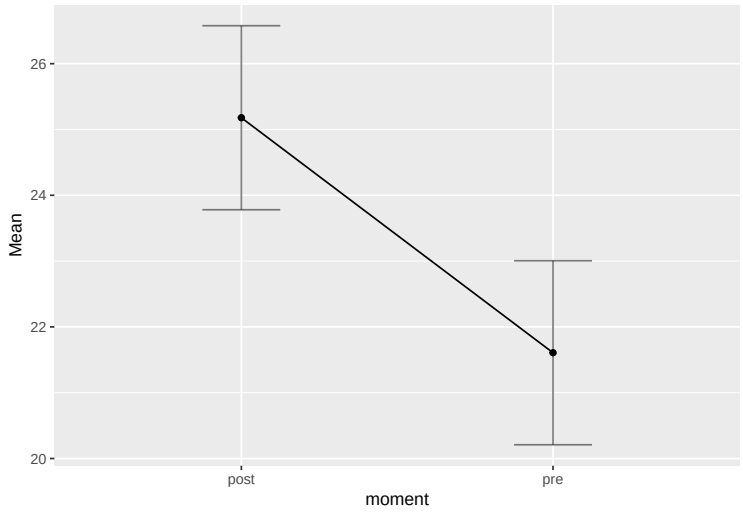
```
describeBy(mixtem$HN, mixtem$moment, mat = T)
```

```
##     item group1 vars  n     mean       sd median  trimmed     mad min max
## X11    1   post    1 56 25.17857 9.877628     23 24.17391 11.8608  14  50
## X12    2    pre    1 56 21.60714 7.325370     20 20.80435  7.4130  14  39
##     range      skew   kurtosis        se
## X11    36 0.6678795 -0.4903363 1.3199535
## X12    25 0.7723564 -0.5299175 0.9788937
```

La représentation graphique peut être réalisée avec la fonction `ezPlot` (paquet `ez`) :

```
ezPlot(data = mixtem, wid = sujet, dv = HN, within = moment, x = moment, do_lines = T)
```

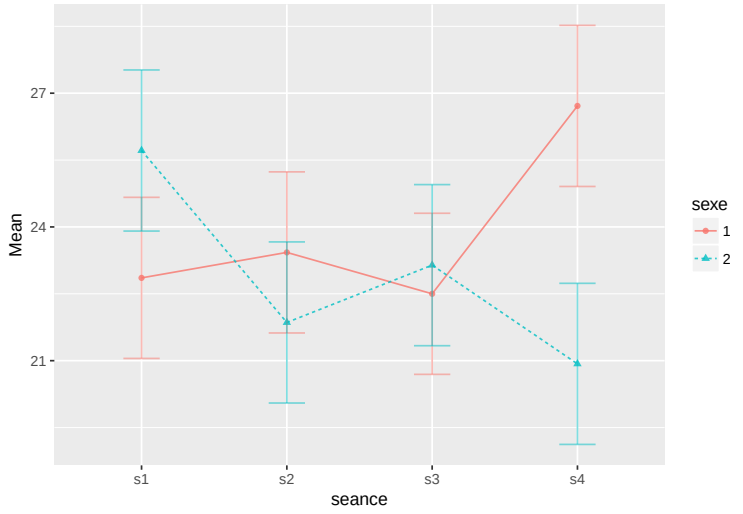
```
## Warning: Collapsing data to cell means. *IF* the requested effects are a
## subset of the full design, you must use the "within_full" argument, else
## results may be inaccurate.
```



En ce qui concerne l'interaction entre la séance et le sexe, on peut visualiser graphiquement l'interaction (toujours avec la même fonction) :

```
ezPlot(data = mixtem, wid = sujet, dv = HN, between = sexe, within = seance,
       x = seance, split = sexe, do_lines = T)
```

```
## Warning: Collapsing data to cell means. *IF* the requested effects are a
## subset of the full design, you must use the "within_full" argument, else
## results may be inaccurate.
```



Graphiquement, on a l'impression que les femmes ont une humeur négative qui diminue au cours des sessions, et notamment entre la 1ère et la 4e séance, alors que chez les hommes, ce serait le contraire. Toutefois, les différences observées ne sont statistiquement significatives que chez les femmes :

```
t.test (mixtem$HN [mixtem$seance == "s1" & mixtem$sexe == "1"],
        mixtem$HN [mixtem$seance == "s4" & mixtem$sexe == "1"], paired = T)
```

```
##
##  Paired t-test
##
## data:  mixtem$HN[mixtem$seance == "s1" & mixtem$sexe == "1"] and mixtem$HN[mixtem$seance == "s4" & mi
## t = -1.7084, df = 13, p-value = 0.1113
```

```
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -8.734666  1.020380
## sample estimates:
## mean of the differences
##               -3.857143
```

```
t.test (mixtem$HN [mixtem$seance == "s1" & mixtem$sexe == "2"],
        mixtem$HN [mixtem$seance == "s4" & mixtem$sexe == "2"], paired = T)
```

```
##
##  Paired t-test
##
## data:  mixtem$HN[mixtem$seance == "s1" & mixtem$sexe == "2"] and mixtem$HN[mixtem$seance == "s4" & mi
## t = 4.0804, df = 13, p-value = 0.0013
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  2.251915 7.319513
## sample estimates:
## mean of the differences
##                4.785714
```

Chapter 12

Régression multiple

Traditionnellement, la régression multiple est utilisée lorsqu'il y a une variable dépendante quantitative et une ou plusieurs variables indépendantes elles aussi quantitatives. On verra toutefois que le modèle linéaire (*linear model*) permet de prendre en compte des variables nominales.

C'est le fichier "employes.csv" (téléchargement) qui servira à illustrer l'utilisation du modèle linéaire avec `R`. Il contient les variables suivantes :

```
emp <- read.csv2("employes.csv")
str (emp)
```

```
## 'data.frame':    474 obs. of  9 variables:
##  $ sujet     : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ sexe      : Factor w/ 2 levels "Féminin","Masculin": 2 2 1 1 2 2 2 1 1 1 ...
##  $ educ      : int  15 16 12 8 15 15 15 12 15 12 ...
##  $ catemp    : Factor w/ 3 levels "Cadre","Responsable",..: 2 3 3 3 3 3 3 3 3 3 ...
##  $ salact    : int  57000 40200 21450 21900 45000 32100 36000 21900 27900 24000 ...
##  $ saldeb    : int  27000 18750 12000 13200 21000 13500 18750 9750 12750 13500 ...
##  $ anciennete: int  98 98 98 98 98 98 98 98 98 98 ...
##  $ exp       : int  144 36 381 190 138 67 114 12 115 244 ...
##  $ minorite  : Factor w/ 2 levels "Non","Oui": 1 1 1 1 1 1 1 1 1 1 ...
```

Parmi celles-ci, la variable "salact" renvoie au salaire actuel (salaire annuel brut), "saldeb" (au salaire lors de l'embauche), "educ" au niveau d'éducation (en nombre d'années à partir du début de la scolarité obligatoire), "anciennete" à l'ancienneté (en mois) dans l'entreprise et "exp" à l'expérience (en mois) dans la fonction.

12.1 Régression avec une variable indépendante quantitative

12.1.1 Le modèle

C'est la fonction `lm` qu'il faut utiliser pour estimer un modèle linéaire. De nombreux résultats sont générés automatiquement à partir de cette fonction qui seront détaillés au fur et à mesure de l'avancement de ce chapitre. C'est pourquoi il est préférable de stocker les résultats dans un objet (appelé `m1` dans l'exemple suivant). Dans la commande qui suit, le salaire actuel est la variable dépendante et le salaire d'embauche la variable indépendante quantitative :

```
m1 <- lm (salact ~ saldeb, data = emp)
str (m1)
```

```
## List of 12
##  $ coefficients : Named num [1:2] 1928.21 1.91
##   ..- attr(*, "names")= chr [1:2] "(Intercept)" "saldeb"
##  $ residuals    : Named num [1:474] 3517 2470 -3392 -5233 2973 ...
##   ..- attr(*, "names")= chr [1:474] "1" "2" "3" "4" ...
##  $ effects      : Named num [1:474] -749367 -326850 -3473 -5330 2771 ...
##   ..- attr(*, "names")= chr [1:474] "(Intercept)" "saldeb" "" "" ...
##  $ rank         : int 2
##  $ fitted.values: Named num [1:474] 53483 37730 24842 27133 42027 ...
##   ..- attr(*, "names")= chr [1:474] "1" "2" "3" "4" ...
##  $ assign       : int [1:2] 0 1
##  $ qr           :List of 5
##   ..$ qr   : num [1:474, 1:2] -21.7715 0.0459 0.0459 0.0459 0.0459 ...
##   .. ..- attr(*, "dimnames")=List of 2
##   .. .. ..$ : chr [1:474] "1" "2" "3" "4" ...
##   .. .. ..$ : chr [1:2] "(Intercept)" "saldeb"
##   .. ..- attr(*, "assign")= int [1:2] 0 1
##   ..$ qraux: num [1:2] 1.05 1.01
##   ..$ pivot: int [1:2] 1 2
##   ..$ tol  : num 1e-07
##   ..$ rank : int 2
##   ..- attr(*, "class")= chr "qr"
##  $ df.residual  : int 472
##  $ xlevels      : Named list()
##  $ call         : language lm(formula = salact ~ saldeb, data = emp)
##  $ terms        :Classes 'terms', 'formula'  language salact ~ saldeb
##   .. ..- attr(*, "variables")= language list(salact, saldeb)
##   .. ..- attr(*, "factors")= int [1:2, 1] 0 1
##   .. .. ..- attr(*, "dimnames")=List of 2
##   .. .. .. ..$ : chr [1:2] "salact" "saldeb"
##   .. .. .. ..$ : chr "saldeb"
##   .. ..- attr(*, "term.labels")= chr "saldeb"
##   .. ..- attr(*, "order")= int 1
##   .. ..- attr(*, "intercept")= int 1
##   .. ..- attr(*, "response")= int 1
##   .. ..- attr(*, ".Environment")=<environment: R_GlobalEnv>
##   .. ..- attr(*, "predvars")= language list(salact, saldeb)
##   .. ..- attr(*, "dataClasses")= Named chr [1:2] "numeric" "numeric"
##   .. .. ..- attr(*, "names")= chr [1:2] "salact" "saldeb"
##  $ model        :'data.frame':   474 obs. of  2 variables:
##   ..$ salact: int [1:474] 57000 40200 21450 21900 45000 32100 36000 21900 27900 24000 ...
##   ..$ saldeb: int [1:474] 27000 18750 12000 13200 21000 13500 18750 9750 12750 13500 ...
##   ..- attr(*, "terms")=Classes 'terms', 'formula'  language salact ~ saldeb
##   .. .. ..- attr(*, "variables")= language list(salact, saldeb)
##   .. .. ..- attr(*, "factors")= int [1:2, 1] 0 1
##   .. .. .. ..- attr(*, "dimnames")=List of 2
##   .. .. .. .. ..$ : chr [1:2] "salact" "saldeb"
##   .. .. .. .. ..$ : chr "saldeb"
##   .. .. ..- attr(*, "term.labels")= chr "saldeb"
##   .. .. ..- attr(*, "order")= int 1
##   .. .. ..- attr(*, "intercept")= int 1
##   .. .. ..- attr(*, "response")= int 1
##   .. .. ..- attr(*, ".Environment")=<environment: R_GlobalEnv>
##   .. .. ..- attr(*, "predvars")= language list(salact, saldeb)
```

```
##   .. .. ..- attr(*, "dataClasses")= Named chr [1:2] "numeric" "numeric"
##   .. .. .. ..- attr(*, "names")= chr [1:2] "salact" "saldeb"
##  - attr(*, "class")= chr "lm"
```

L'objet de résultats `m1` est donc une liste contenant 12 éléments. Pour extraire les résultats principaux, on utilise la fonction `summary` :

```
summary (m1)
```

```
##
## Call:
## lm(formula = salact ~ saldeb, data = emp)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -35424  -4031  -1154   2584  49293
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 1.928e+03  8.887e+02    2.17   0.0305 *
## saldeb      1.909e+00  4.741e-02   40.28   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 8115 on 472 degrees of freedom
## Multiple R-squared:  0.7746, Adjusted R-squared:  0.7741
## F-statistic:  1622 on 1 and 472 DF,  p-value: < 2.2e-16
```

Ces résultats fournissent les coefficients de régression qui permettent d'écrire l'équation de régression :

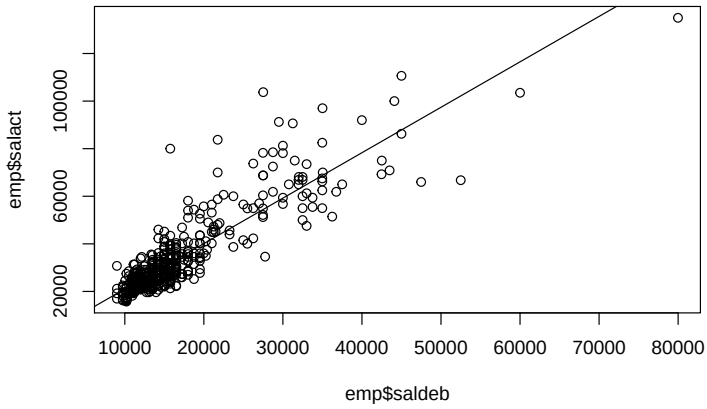
$$salact = 1928 + 1.909 \times saldeb$$

L'intercept (ou constante) désigne la valeur de la variable dépendante quand la variable dépendante est égale à zéro (soit 1929 dans ce modèle, ce qui n'est pas vraiment utilisable ici, étant donné que les valeurs observées de la variable indépendante sont très éloignées de zéro). Le coefficient de régression de "saldeb" est égal à 1.909 ; il correspond à la pente de la droite de régression et indique donc que pour une différence de 1 euro dans le salaire d'embauche, la différence de salaire actuel est en moyenne de 1.909 euros (soit près de deux fois plus).

12.1.2 Représentation graphique

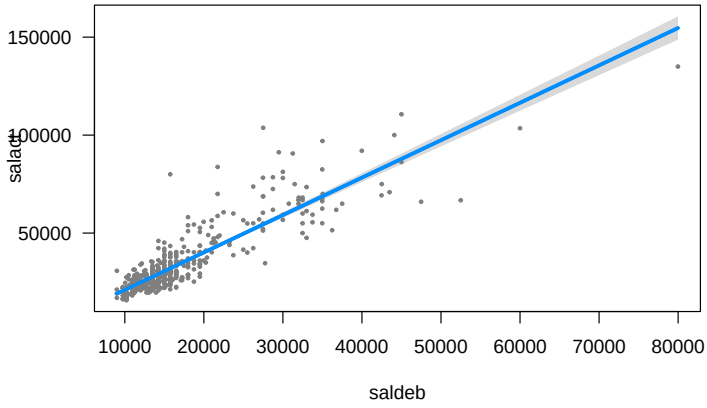
L'équation de régression correspond donc à une droite dans le cas d'une seule variable indépendante et celle-ci peut-être représentée graphiquement à l'aide de la fonction générique `plot` (pour le nuage de points) à laquelle s'ajoute la fonction `abline` qui dessine une droite en fonction des valeurs des coefficients du modèle

```
par (mfrow = c(1,1))
plot (emp$saldeb, emp$salact)
abline (m1)
```



Plusieurs paquets permettent de construire automatiquement des graphiques de meilleure qualité, avec des options très utiles lorsque les modèles deviennent plus complexes. C'est le paquet `visreg` (Breheny & Burchett (2017)) qui sera utilisé dans ce chapitre. La fonction principale est éponyme, comme le montre l'exemple ci-dessous. La partie grisée autour de la droite de régression permet de visualiser son intervalle de confiance.

```
library (visreg)
visreg (m1)
```



Le nuage de points n'est pas très régulier et montre une très grande concentration des observations autour de valeurs faibles des deux variables. On reviendra sur cette particularité un peu plus loin.

12.1.3 Les coefficients standardisés

La fonction `lm` fournit les coefficients non standardisés et il n'existe pas d'options dans cette fonction pour obtenir les coefficients standardisés. Mais comme les coefficients standardisés sont simplement les coefficients d'une régression dans laquelle la variance de toutes les variables a été ramenée à 1, il suffit de faire une régression avec les variables centrées-réduites pour obtenir les coefficients standardisés. La fonction `scale` permet de centrer-réduire une variable. Ainsi, on peut écrire :

```
summary (lm (scale(salact) ~ scale(saldeb), data = emp))
```

```
##
## Call:
## lm(formula = scale(salact) ~ scale(saldeb), data = emp)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.0745 -0.2360 -0.0676  0.1514  2.8867
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)   -8.693e-17  2.183e-02    0.00        1
## scale(saldeb)  8.801e-01  2.185e-02   40.28   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4753 on 472 degrees of freedom
## Multiple R-squared:  0.7746, Adjusted R-squared:  0.7741
## F-statistic:  1622 on 1 and 472 DF,  p-value: < 2.2e-16
```

Une autre manière de procéder consiste à utiliser la fonction `lm.beta` du paquet éponyme (Behrendt (2014)). Cette fonction s'applique directement sur l'objet-résultat issu de la fonction `lm` :

```
library(lm.beta)
lm.beta(m1)
```

```
##
## Call:
## lm(formula = salact ~ saldeb, data = emp)
##
## Standardized Coefficients::
## (Intercept)      saldeb
##   0.0000000   0.8801175
```

On peut aussi combiner la fonction `lm.beta` avec la fonction `summary` pour avoir un tableau de résultat avec en même temps les coefficients standardisés et les coefficients non standardisés :

```
summary (lm.beta(m1))
```

```
##
## Call:
## lm(formula = salact ~ saldeb, data = emp)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -35424  -4031  -1154   2584  49293
##
## Coefficients:
##              Estimate Standardized Std. Error t value Pr(>|t|)
## (Intercept) 1.928e+03    0.000e+00  8.887e+02    2.17   0.0305 *
## saldeb      1.909e+00    8.801e-01  4.741e-02   40.28   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 8115 on 472 degrees of freedom
```

```
## Multiple R-squared:  0.7746, Adjusted R-squared:  0.7741
## F-statistic:  1622 on 1 and 472 DF,  p-value: < 2.2e-16
```

12.1.4 Les valeurs prédites et les résidus

L'équation de régression prédit un salaire actuel en fonction du salaire d'embauche. Cette prédiction n'est jamais parfaite avec des données empiriques et les résidus expriment l'écart qu'il y a entre la valeur réelle de la variable dépendante et la valeur prédite. L'élément `fitted.values` de l'objet-résultat contient les valeurs prédites et l'élément `residuals` les résidus. Les résidus standardisés sont, quant à eux, disponibles grâce à la fonction `rstandard`

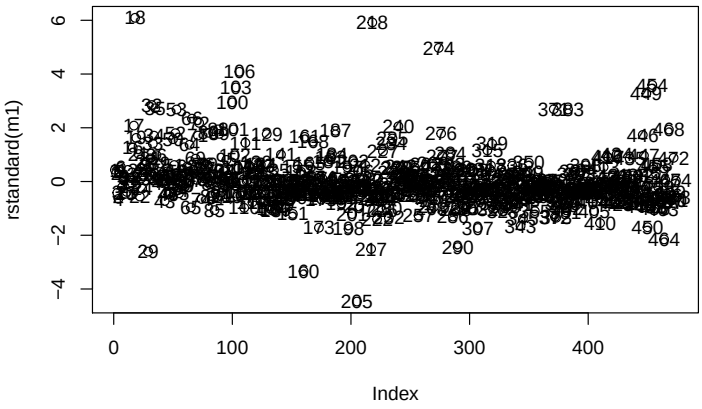
Dans la ligne de commande ci-dessous, on fait apparaître (pour les 15 premiers sujets uniquement) les valeurs réelles du salaire actuelle, les valeurs prédites, les résidus et les résidus standardisés :

```
data.frame ("salaire actuel réel" = emp$salact, "salaire prédit" = m1$fitted.values,
            "résidu" = m1$residuals, "résidus standardisés" = rstandard (m1)) [1:15,]
```

```
##    salaire.actuel.réel salaire.prédit     résidu résidus.standardisés
## 1                57000       53483.35  3516.6516           0.43453216
## 2                40200       37730.39  2469.6118           0.30465062
## 3                21450       24841.60 -3391.6025          -0.41854575
## 4                21900       27132.94 -5232.9422          -0.64566190
## 5                45000       42026.65  2973.3499           0.36687234
## 6                32100       27705.78  4394.2229           0.54215686
## 7                36000       37730.39 -1730.3882          -0.21346020
## 8                21900       20545.34  1354.6594           0.16725288
## 9                27900       26273.69  1626.3102           0.20067332
## 10               24000       27705.78 -3705.7771          -0.45721679
## 11               30300       33434.13 -3134.1263          -0.38660682
## 12               28350       24841.60  3508.3975           0.43295901
## 13               27750       29137.86 -1387.8644          -0.17122016
## 14               35100       34006.96  1093.0388           0.13483013
## 15               27300       27705.78  -405.7771          -0.05006456
```

Une représentation graphique permet de repérer facilement les sujets avec des résidus standardisés les plus importants (qui ne se conforment donc pas au modèle) :

```
plot (rstandard (m1))
text (rstandard (m1), labels = emp$sujet)
```



12.1.5 Les observations influentes

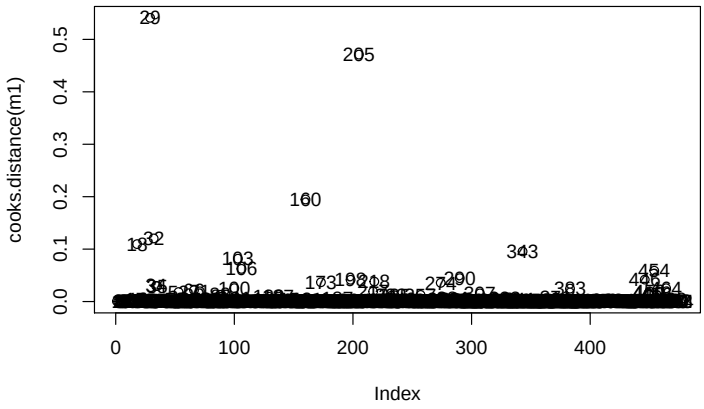
Outre les sujets ne se conformant pas au modèle, il peut y avoir certains sujets qui ont une forte influence sur les résultats de la régression, soit parce qu'il s'agit de sujets réellement hors normes, soit, plus prosaïquement, parce que le fichier comporte des erreurs de codage. Les *distances de Cook* sont une manière traditionnelle de quantifier l'impact d'un sujet sur les coefficients de régression. Ils sont obtenus avec la fonction `cooksdistance` :

```
cooks.distance (m1) [1:15]  # seulement pour les 15 premiers sujets
```

```
##            1            2            3            4            5
## 5.232287e-04 1.028921e-04 2.607788e-04 5.447604e-04 1.789060e-04
##            6            7            8            9           10
## 3.730108e-04 5.051401e-05 5.492491e-05 5.513550e-05 2.652869e-04
##           11           12           13           14           15
## 1.586789e-04 2.790487e-04 3.483463e-05 1.923142e-05 3.180770e-06
```

Ici, aussi, un graphique est utile pour repérer rapidement les sujets concernés par des distances de Cook élevées :

```
plot (cooks.distance (m1))
text (cooks.distance (m1), labels = emp$sujet)
```



12.1.6 Les postulats de la régression

La normalité des résidus et l'homoscédasticité sont deux postulats importants de la régression.

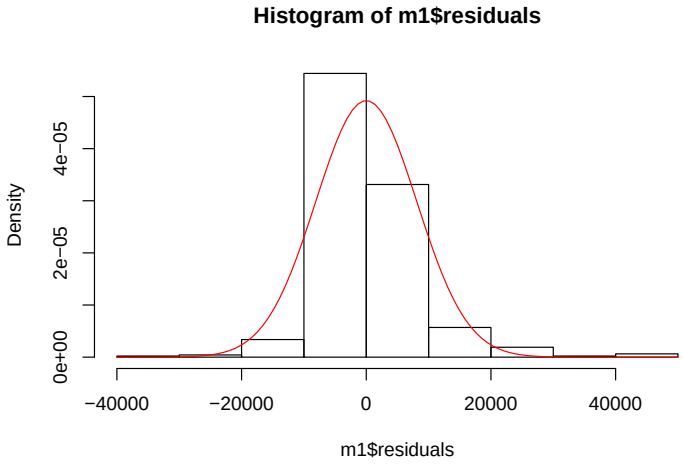
12.1.6.1 Normalité des résidus

La normalité des résidus du modèle `m1` est problématique comme le montrent les résultats ci-dessous :

```
library (psych)
describe (m1$residuals)
```

```
##    vars   n mean      sd   median trimmed     mad       min      max
## X1    1 474    0 8106.77 -1154.31 -779.75 4770.88 -35424.32 49292.83
##       range skew kurtosis     se
## X1 84717.15 1.62     8.14 372.36
```

```
hist(m1$residuals, prob=T)
curve(dnorm (x, mean=mean(m1$residuals), sd=sd(m1$residuals)), col=2, add = TRUE)
```



On suspecte que le problème, ici, vient de la non-normalité de la distribution du salaire. Le résultat suivant montre que la normalité des résidus s'améliore lorsque l'on prend le logarithme du salaire actuel :

```
m2 <- lm (log(salact) ~ saldeb, data = emp)
describe (m2$residuals)
```

```
##    vars   n mean   sd median trimmed  mad   min  max range skew kurtosis
## X1    1 474    0 0.22  -0.01   -0.01 0.19 -1.22 0.99   2.2 0.14     3.17
##      se
## X1 0.01
```

En prenant aussi le logarithme du salaire d'embauche (modèle `m3`), la normalité des résidus devient tout à fait acceptable :

```
m3 <- lm (log(salact) ~ log (saldeb), data = emp)
describe (m3$residuals)
```

```
##    vars   n mean   sd median trimmed  mad   min  max range skew kurtosis
## X1    1 474    0 0.18  -0.01   -0.01 0.18 -0.47 0.94   1.4 0.66     1.43
##      se
## X1 0.01
```

Bien évidemment, en transformant la distribution des variables, les résultats de la régression ne sont plus les mêmes : les coefficients non standardisés changent, mais aussi les coefficients non standardisés et le coefficient de détermination :

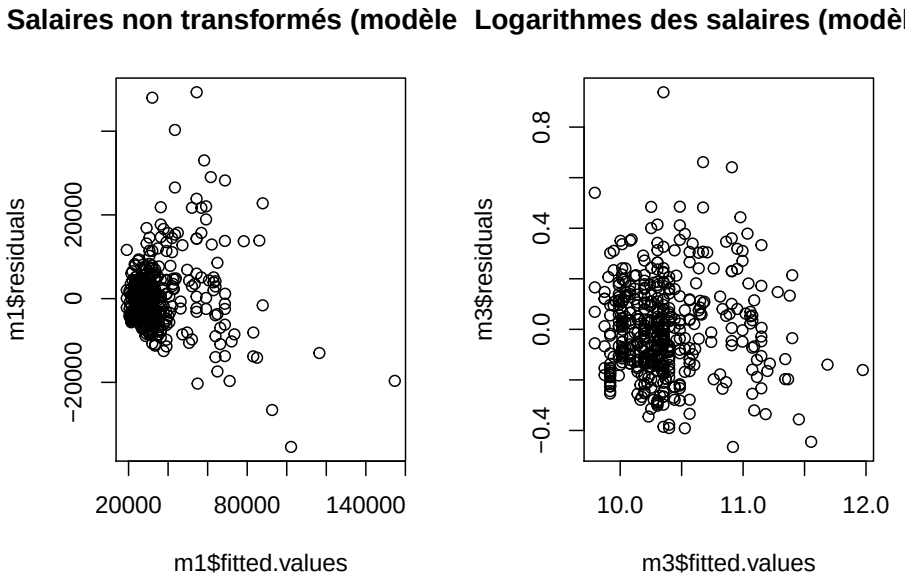
```
summary (lm.beta(m3))
```

```
##
## Call:
## lm(formula = log(salact) ~ log(saldeb), data = emp)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.46515 -0.12758 -0.01248  0.10985  0.93779
##
## Coefficients:
##             Estimate Standardized Std. Error t value Pr(>|t|)
## (Intercept)   0.7054       0.0000     0.2322   3.038  0.00251 **
## log(saldeb)   0.9981       0.8864     0.0240  41.593  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1842 on 472 degrees of freedom
## Multiple R-squared:  0.7856, Adjusted R-squared:  0.7852
## F-statistic:  1730 on 1 and 472 DF,  p-value: < 2.2e-16
```

12.1.6.2 Homoscédasticité

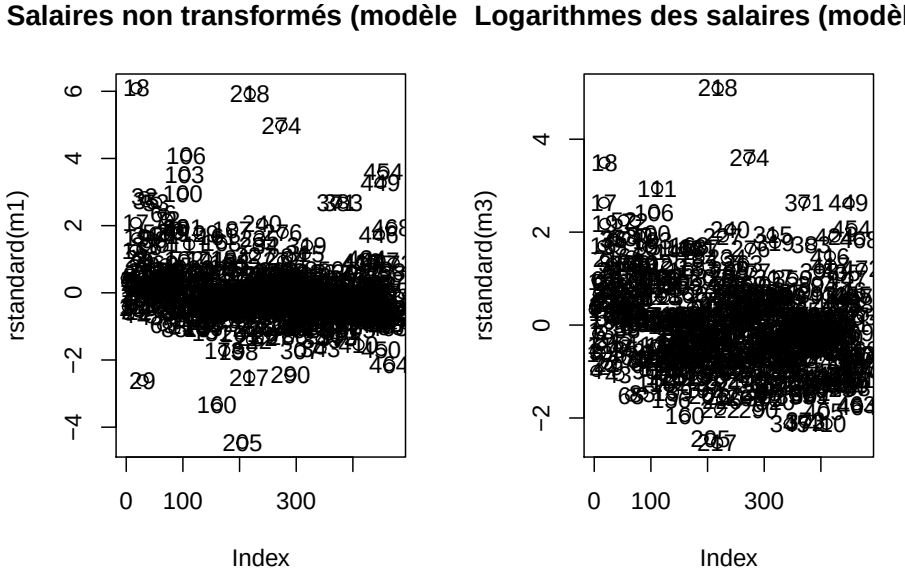
L'homoscédasticité requiert que les résidus aient une variance constante, quel que soit le niveau des valeurs prédites de la variable dépendante. Ce n'est pas vraiment le cas ici (figure du haut : modèle `m1`), mais là aussi, l'analyse sur le logarithme des salaires permet de réduire de manière sensible l'hétéroscédasticité (figure du bas : modèle `m3`) :

```
par (mfrow = c(1,2))
plot (m1$fitted.values, m1$residuals, main = "Salaires non transformés (modèle m1)")
plot (m3$fitted.values, m3$residuals, main = "Logarithmes des salaires (modèle m3)")
```



Ces transformations concourrent aussi à faire diminuer les résidus les plus extrêmes que l'on observait dans le modèle `m1` :

```
par (mfrow = c(1,2))
plot (rstandard (m1), main = "Salaires non transformés (modèle m1)")
text (rstandard (m1), labels = emp$sujet)
plot (rstandard (m3), main = "Logarithmes des salaires (modèle m3)")
text (rstandard (m3), labels = emp$sujet)
```



Dans la suite du chapitre, la vérification des postulats de la régression et l'analyse des résidus ne sera plus conduite systématiquement pour ne pas augmenter le nombre de pages du document de manière inconsidérée. Il est toutefois indispensable de procéder à ces vérifications, selon la logique exposée dans cette section.

12.2 Régression avec deux variables indépendantes quantitatives

12.2.1 Le modèle

Avec plusieurs variables indépendantes, la régression offre la possibilité d'interpréter les coefficients "toutes choses égales par ailleurs" (*ceteribus paribus*). L'exemple suivant, d'abord avec une seule variable indépendante (l'expérience) puis avec deux (l'expérience et le niveau d'éducation) montre l'importance de cette propriété. En effet, on s'aperçoit que, dans le modèle `m4`, le coefficient de régression de l'expérience est négatif, alors qu'il devient positif dans le modèle `m5`.

```
m4 <- lm (log (salact) ~ exp, data = emp)
summary (lm.beta (m4))
```

```
##
## Call:
## lm(formula = log(salact) ~ exp, data = emp)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.70881 -0.26088 -0.09758  0.16533  1.50626
##
## Coefficients:
##               Estimate Standardized Std. Error t value Pr(>|t|)
## (Intercept) 10.4038597    0.0000000  0.0247202 420.865  < 2e-16 ***
## exp         -0.0004879   -0.1277787  0.0001743  -2.799  0.00534 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3945 on 472 degrees of freedom
## Multiple R-squared:  0.01633,    Adjusted R-squared:  0.01424
## F-statistic: 7.834 on 1 and 472 DF,  p-value: 0.005336
```

```
m5 <- lm (log (salact) ~ exp + educ, data = emp)
summary (lm.beta (m5))
```

```
##
## Call:
## lm(formula = log(salact) ~ exp + educ, data = emp)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.65666 -0.18986 -0.02733  0.17094  0.95291
##
## Coefficients:
##              Estimate Standardized Std. Error t value Pr(>|t|)
## (Intercept) 9.0160429    0.0000000  0.0689453 130.771   <2e-16 ***
## exp         0.0002081    0.0545106  0.0001302   1.598    0.111
## educ        0.0978887    0.7107212  0.0046972  20.840   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2849 on 471 degrees of freedom
## Multiple R-squared:  0.4882, Adjusted R-squared:  0.486
```

```
## F-statistic: 224.7 on 2 and 471 DF,  p-value: < 2.2e-16
```

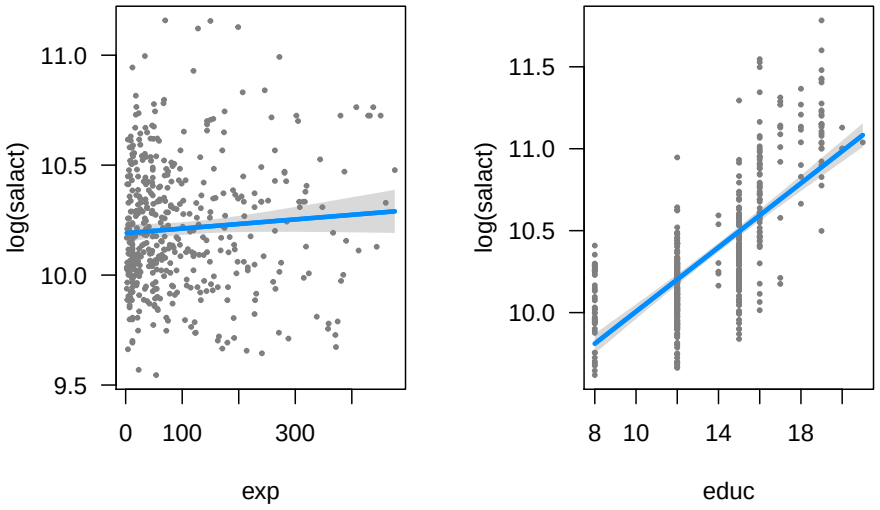
L'équation de régression s'écrit donc de la manière suivante :

$$salact = 9.016 + 0.0002 \times exp + 0.0979 \times educ$$

La représentation graphique d'un modèle de régression avec plusieurs variables indépendantes nécessite autant de dimensions qu'il y a de variables (variable dépendante + variables indépendantes). Avec trois variables, il est encore possible de représenter le plan de régression en trois dimensions, mais cela devient impossible avec plus de trois variables.

Appliquée à l'objet-résultat `m5`, la fonction `visreg` propose deux graphiques :

```
par (mfrow = c(1,2))
visreg (m5)
```



Le premier graphique permet de visualiser l'effet de l'expérience et le second celui du niveau d'éducation. Pour construire ces deux droites, la valeur de l'autre variable est fixée à sa médiane.

12.2.2 La multicolinéarité

Il est toujours possible d'augmenter le pourcentage de variance expliquée (i.e. le coefficient de détermination) de la variable dépendante en ajoutant des variables indépendantes dans le modèle. Toutefois, si les variables indépendantes dépendent trop fortement les unes des autres (i.e. si l'une est la combinaison linéaire presque parfaite des autres), il y a *multicolinéarité*. Cette situation entraîne une grande instabilité des coefficients estimés et une augmentation très importante des erreurs standards. Le *VIF* (*Variance Inflation Factor*) est un indicateur souvent utilisé pour évaluer le risque de multicolinéarité[1]. Il s'obtient avec la fonction `vif` du paquet `faraway` (Faraway (2016))[2] :

[1] $VIF_j = \frac{1}{1-R_j^2}$ Autrement dit le VIF pour une variable est dépend du coefficient de détermination de la régression de cette variable en fonction de toutes les autres variables indépendantes. Des VIF supérieurs à 5 commencent à être considérés comme problématiques.

[2] Une fonction `vif` est disponible dans d'autres paquets, notamment le paquet `car` (Fox & Weisberg (2011)).

```
library (faraway)
```

```
## Warning: le package 'faraway' a été compilé avec la version R 3.4.3
```

```
##
## Attachement du package : 'faraway'
```

```
## The following object is masked from 'package:psych':
##
##     logit
```

```
vif (m5)
```

```
##      exp     educ
## 1.070417 1.070417
```

Il n'y a aucun problème avec ces deux variables indépendantes. Mais comme on peut le voir ci-dessous, les *VIF* commencent à augmenter (sans causer d'inquiétudes particulières) si on ajoute le salaire d'embauche dans le modèle[3] :

```
vif (lm (salact ~ saldeb + educ + exp, data = emp))
```

```
##   saldeb     educ      exp
## 1.802723 1.926400 1.155992
```

12.3 Régression non linéaire

L"appellation *modèle linéaire* peut laisser penser que celui-ci ne peut rendre compte que des relations linéaires qui existent entre la variable dépendante et les variables indépendantes. Cela n'est pas tout à fait vrai et il suffit d'une petite astuce pour prendre en compte des éventuels effets non linéaires. Si on reprend le modèle `m1` où la variable dépendante est le salaire actuel et la variable indépendante le salaire d'embauche, on a l'équation de régression suivante : $salact = b_0 + b_1 \times saldeb$. Il s'agit bien de l'équation d'une droite. Mais si on ajoute au modèle le carré du salaire d'embauche et/ou le cube du salaire d'embauche, l'équation ne forme plus une droite, mais prend une forme curvilinéaire. L'équation de régression devient donc : $salact = b_0 + b_1 \times saldeb + b_1 \times saldeb^2 + b_3 \times saldeb^3$. Elle est estimée dans le modèle `m6`[4] :

```
m6 <- lm (salact ~ saldeb + I(saldeb**2) + I(saldeb**3) , data = emp)
summary (lm.beta (m6))
```

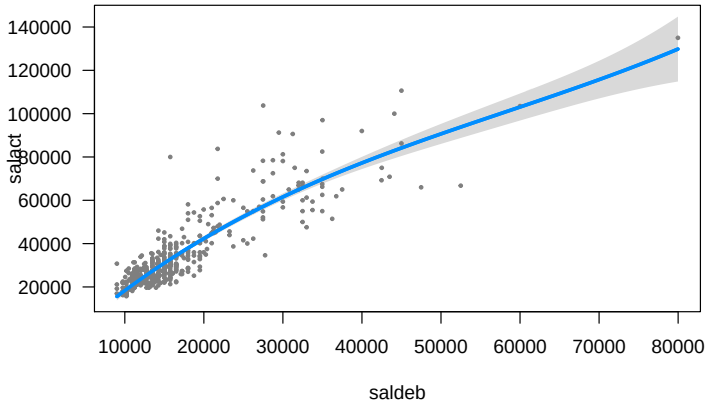
```
##
## Call:
## lm(formula = salact ~ saldeb + I(saldeb^2) + I(saldeb^3), data = emp)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -27061  -4129   -904   3080  47214
##
## Coefficients:
##               Estimate Standardized Std. Error t value Pr(>|t|)
## (Intercept) -1.173e+04    0.000e+00  3.556e+03  -3.300  0.00104 **
## saldeb       3.350e+00    1.544e+00  4.092e-01   8.187 2.54e-15 ***
## I(saldeb^2) -3.651e-05   -1.004e+00  1.292e-05  -2.825  0.00493 **
## I(saldeb^3)  2.096e-10    3.580e-01  1.126e-10   1.862  0.06327 .
```

[3] Cette augmentation des VIF est due essentiellement au fait qu'il y a une corrélation substantielle entre deux des trois variables indépendantes : le salaire d'embauche et le niveau d'éducation.

[4] Noter bien l'utilisation de la fonction `I` pour indiquer les polynomes de la variable "saldeb".

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 7859 on 470 degrees of freedom
## Multiple R-squared:  0.7895, Adjusted R-squared:  0.7882
## F-statistic: 587.7 on 3 and 470 DF,  p-value: < 2.2e-16
```

```
par (mfrow = c(1,1))
visreg (m6)
```



Les résultats font apparaître des effets significatifs du polynome d'ordre 2, mais pas du polynome d'ordre 3. La représentation graphique illustre la forme curvilinéaire de la relation. Mais en introduisant dans le modèle directement les polynomes de la variable dépendante, on crée une situation de forte multicolinéarité comme le montrent les résultats ci-dessous :

```
vif (m6)
```

```
##      saldeb I(saldeb^2) I(saldeb^3)
##    79.42373   282.19120    82.57702
```

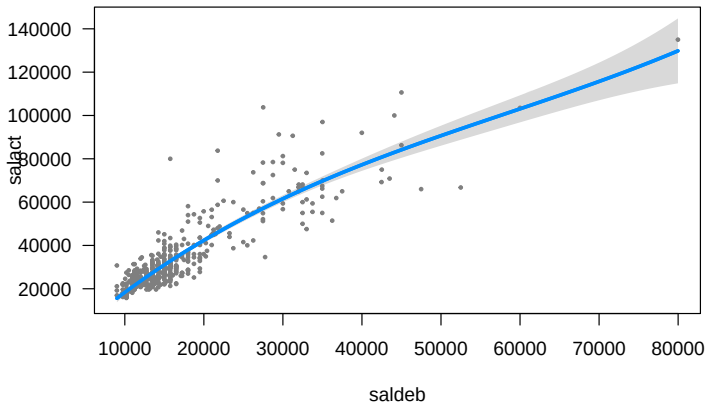
Il est donc préférable d'utiliser la fonction `poly` qui permet de créer des polynomes orthogonaux (i.e. indépendants les uns des autres). L'utilisation de cette fonction est simple, puisqu'il faut simplement indiquer le nom de la variable et le nombre de polynomes souhaités :

```
m7 <- lm (salact ~ poly (saldeb, 3), data = emp)
summary (lm.beta (m7))
```

```
##
## Call:
## lm(formula = salact ~ poly(saldeb, 3), data = emp)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -27061  -4129   -904   3080  47214
##
## Coefficients:
##                      Estimate Standardized Std. Error t value Pr(>|t|)
## (Intercept)       3.442e+04    0.000e+00  3.610e+02  95.353  < 2e-16 ***
## poly(saldeb, 3)1  3.269e+05    8.801e-01  7.859e+03  41.590  < 2e-16 ***
## poly(saldeb, 3)2 -4.294e+04   -1.156e-01  7.859e+03  -5.464 7.58e-08 ***
```

```
## poly(saldeb, 3)3  1.463e+04    3.940e-02  7.859e+03   1.862   0.0633 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 7859 on 470 degrees of freedom
## Multiple R-squared:  0.7895, Adjusted R-squared:  0.7882
## F-statistic: 587.7 on 3 and 470 DF,  p-value: < 2.2e-16
```

```
visreg (m7)
```

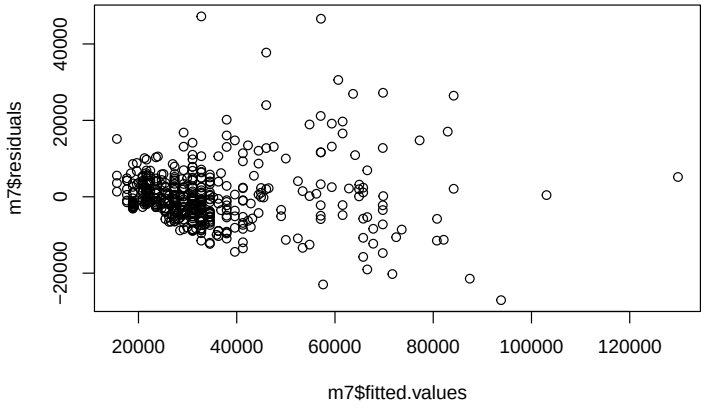


Il faut toutefois noter dans ces résultats une fin de courbe peu plausible puisque celle-ci semble se redresser et conduire à des élévations de salaires quasi infinies. On l'a vu auparavant, les postulats de la régression ne sont pas respectées en utilisant les salaires non transformés et c'est encore le cas ici :

```
describe (m7$residuals)  # normalité des résidus
```

```
##    vars   n mean      sd  median trimmed     mad       min      max
## X1    1 474    0 7833.87 -904.29 -607.84 5064.43 -27061.07 47214.27
##       range skew kurtosis     se
## X1 74275.34 1.55     7.24 359.82
```

```
plot (m7$fitted.values, m7$residuals)  # homoscédasticité
```

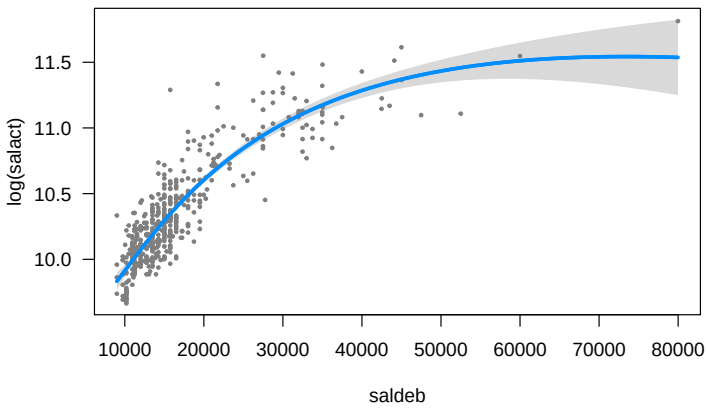


En introduisant les logarithmes des salaires, on obtient les résultats suivants :

```
m8 <- lm (log (salact) ~ poly (log (saldeb), 3), data = emp)
summary (lm.beta (m8))
```

```
##
## Call:
## lm(formula = log(salact) ~ poly(log(saldeb), 3), data = emp)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.50103 -0.11978 -0.00725  0.10743  0.94329
##
## Coefficients:
##                          Estimate Standardized Std. Error  t value Pr(>|t|)
## (Intercept)            10.356793     0.000000   0.008388 1234.743  < 2e-16
## poly(log(saldeb), 3)1   7.659501     0.886368   0.182616   41.943  < 2e-16
## poly(log(saldeb), 3)2 -0.240761    -0.027861   0.182616   -1.318  0.18801
## poly(log(saldeb), 3)3 -0.524318    -0.060675   0.182616   -2.871  0.00427
##
## (Intercept)            ***
## poly(log(saldeb), 3)1 ***
## poly(log(saldeb), 3)2
## poly(log(saldeb), 3)3 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1826 on 470 degrees of freedom
## Multiple R-squared:  0.7901, Adjusted R-squared:  0.7888
## F-statistic: 589.7 on 3 and 470 DF,  p-value: < 2.2e-16
```

```
visreg (m8)
```




```
vif (m8)
```

```
## poly(log(saldeb), 3)1 poly(log(saldeb), 3)2 poly(log(saldeb), 3)3
##                     1                     1                     1
```

On notera que le polynome d'ordre 2 de "saldeb" n'est plus significatif (par rapport au modèle `m7`) et que

la forme de la courbe est plus plausible puisqu'on assiste à un certain tassement de celle-ci au niveau des salaires les plus élevés (même si l'intervalle de confiance est large, ce qui est normal étant donné qu'il y a peu de valeurs observées à ces niveaux de salaire).

12.4 Régression avec des variables indépendantes nominales

12.4.1 Une variable indépendante dichotomique

Dans le tableau de données "emp", la variable "sexe" est une variable dichotomique qui est définie comme un facteur :

```
str (emp$sexe)
```

```
##  Factor w/ 2 levels "Féminin","Masculin": 2 2 1 1 2 2 2 1 1 1 ...
```

Si on introduit cette variable dans une régression, on obtient les résultats suivants :

```
m9 <- lm (salact ~ sexe, data = emp)
summary (lm.beta (m9))
```

```
##
## Call:
## lm(formula = salact ~ sexe, data = emp)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -21792 -10054  -3232   3698  93558
##
## Coefficients:
##               Estimate Standardized Std. Error t value Pr(>|t|)
## (Intercept)  2.603e+04    0.000e+00  1.039e+03   25.06   <2e-16 ***
## sexeMasculin 1.541e+04    4.499e-01  1.408e+03   10.95   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 15270 on 472 degrees of freedom
## Multiple R-squared:  0.2024, Adjusted R-squared:  0.2007
## F-statistic: 119.8 on 1 and 472 DF,  p-value: < 2.2e-16
```

Pour effectuer cette régression, le logiciel R a recodé la variable "sexe" en une variable numérique comprenant deux modalités : 0 pour les femmes et 1 pour les hommes. La modalité "femmes" est la catégorie dite de référence. Elle a été choisie automatiquement parce qu'elle arrive en premier dans l'ordre alphabétique.[5] Comme l'intercept est la valeur de la variable dépendante lorsque les variables indépendantes sont égales à zéro (et qu'il n'y a qu'une seule variable indépendante dans ce modèle), cela signifie que l'intercept représente le salaire moyen des femmes. Le coefficient de régression de la variable "sexe"[6] est égal à 15410 ; il correspond à la différence de salaire moyen entre les femmes et les hommes. Les hommes gagnent en moyenne 15410 euros annuels bruts de plus que les femmes. Pour se convaincre que l'intercept représente le salaire moyen des femmes et le coefficient de régression de la variable "sexe" la différence de salaire moyen entre les hommes et les femmes, il suffit de calculer les salaires moyens en fonction du sexe :

[5] On peut changer l'ordre des différentes modalités d'un facteur. La commande `emp$sexe <- factor (emp$sexe, levels = c("Masculin", "Féminin"))` permet de définir la modalité "Masculin" comme première modalité et donc comme catégorie de référence.

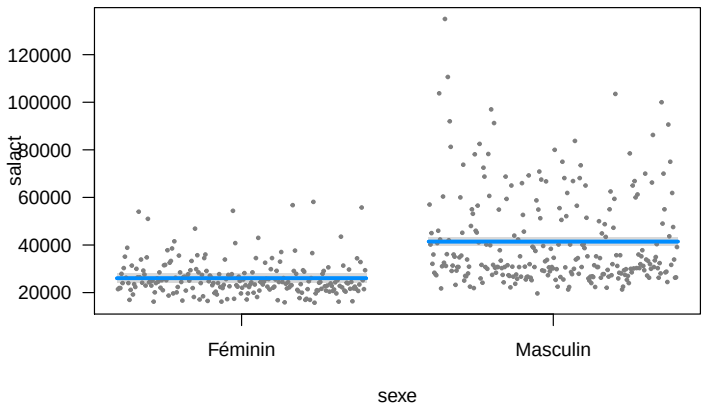
[6] Il est appelé "sexeMasculin" dans le tableau de résultats pour indiquer qu'il s'agit de la différence de la catégorie "Masculin" par rapport à la catégorie de référence ("Féminin").

```
describeBy(emp$salact, emp$sexe, mat = T)
```

```
##     item   group1 vars   n     mean        sd median  trimmed     mad
## X11    1  Féminin    1 216 26031.92  7558.021  24300 25012.76 5114.97
## X12    2 Masculin    1 258 41441.78 19499.214  32850 38256.15 9785.16
##       min    max  range     skew kurtosis        se
## X11 15750  58125  42375 1.837317 4.437483  514.2582
## X12 19650 135000 115350 1.620464 2.658797 1213.9680
```

Le représentation graphique de ce modèle fait bien appaître les deuc moyennes et leurs différences. On notera que la fonction `visreg` construit un graphique qui n'aligne pas verticalement tous les points correspondant aux femmes et tous les points correspondant aux hommes, mais les décale les uns des autres pour fournir une représentation plus "esthétique" :

```
visreg(m9)
```



12.4.2 Une variable indépendante polytomique

L'introduction d'une variable indépendante polytomique (à plus de deux modalités) est permise grâce à une logique similaire. Mais il est nécessaire de passer par l'intermédiaire de variables dites "factices" (en anglais : *dummy variables*). Prenons l'exemple de la variable "catemp" qui comporte trois modalités :

```
table (emp$catemp)
```

```
##
##       Cadre Responsable Secrétariat
##          27          84         363
```

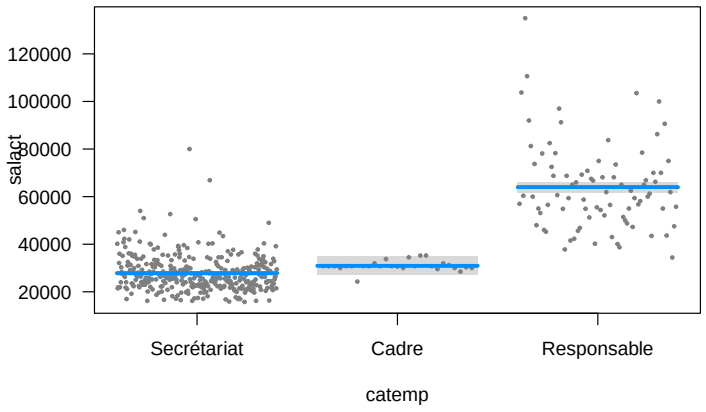
On peut définir la modalité "Cadre" comme catégorie de référence. La catégorie "Responsable" deviendra donc la première variable factice (et son coefficient de régression indiquera la différence de salaire entre les cadres et les responsables) ; la catégorie "Secrétariat" deviendra quant à elle la seconde variable factice. En généralisant, pour une variable nominales à k modalités, il faut construire k-1 variables factices, car l'une des modalités est définie comme catégorie de référence.

Dans l'application qui suit, on change tout d'abord l'ordre des modalités de la variable "catemp" (pour que la modalité "Secrétariat" devienne la catégorie de référence) et on estime les coefficients du modèle `m10` dans lequel le salaire actuel est la variable dépendante et la catégorie d'emploi la variable indépendante :

```
emp$catemp <- factor (emp$catemp, levels = c("Secrétariat", "Cadre", "Responsable"))
m10 <- lm (salact ~ catemp, data = emp)
summary (lm.beta (m10))
```

```
##
## Call:
## lm(formula = salact ~ catemp, data = emp)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -29568  -5339  -1139   3551  71022
##
## Coefficients:
##                    Estimate Standardized Std. Error t value Pr(>|t|)
## (Intercept)       2.784e+04    0.000e+00  5.325e+02  52.280   <2e-16 ***
## catempCadre       3.100e+03    4.213e-02  2.024e+03   1.532    0.126
## catempResponsable 3.614e+04    8.090e-01  1.228e+03  29.421   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 10150 on 471 degrees of freedom
## Multiple R-squared:  0.6485, Adjusted R-squared:  0.647
## F-statistic: 434.5 on 2 and 471 DF,  p-value: < 2.2e-16
```

```
visreg(m10)
```



Il est intéressant de noter que l'analyse de variance (résultats ci-dessous) réalisée à partir des mêmes variables ("salact" et "catemp") fournit strictement les mêmes résultats. La valeur du `F` et le pourcentage de variance expliquée ("Multiple R-squared" dans l'analyse de régression et "ges" dans l'analyse de variance) sont bien identiques :

```
library (ez)
ezANOVA(data = emp, wid = sujet, dv = salact, between = catemp, detailed = T)
```

```
## Warning: Converting "sujet" to factor for ANOVA.
```

```
## Warning: Data is unbalanced (unequal N per group). Make sure you specified
## a well-considered value for the type argument to ezANOVA().
```

```
## $ANOVA
##   Effect DFn DFd         SSn         SSd        F             p p<.05
## 1 catemp   2 471 89438483926 48478011510 434.4808 1.164613e-107     *
##         ges
## 1 0.6484974
##
## $`Levene's Test for Homogeneity of Variance`
##   DFn DFd        SSn         SSd        F            p p<.05
## 1   2 471 5333173157 24535917585 51.18872 7.649994e-21     *
```

On notera aussi que la régression fournit de manière automatique non seulement un effet global de la variable indépendante (comme l'analyse de variance), mais aussi la significativité des différences deux à deux entre les moyennes de la catégorie de référence et celles des autres catégories. Ces comparaisons (ici : secrétariat vs. cadre et secrétariat vs. responsable) sont statistiquement indépendantes et ne nécessitent pas d'ajustement spécifique due aux comparaisons multiples que l'on a discuté dans le chapitre sur l'analyse de variance. La question des comparaisons multiples se pose en revanche si en plus de ces deux comparaisons, on voudrait tester la différence de moyennes entre les cadres et les responsables.

12.4.3 Tester globalement la significativité d'une variable nominale polytomique

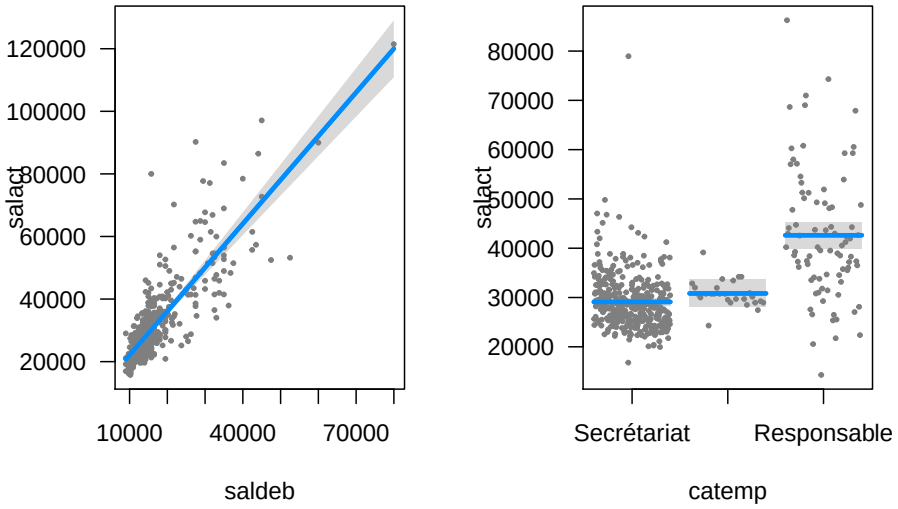
Le modèle suivant (`m11`) contient deux variables indépendantes : le salaire d'embauche et la catégorie d'emploi. Dans l'écriture de la commande, c'est le signe `+` qui est utilisé, ce qui indique que l'on ne cherche à estimer que les effets principaux des deux variables, sans leur interaction[7].

```
m11 <- lm (salact ~ saldeb + catemp, data = emp)
summary (lm.beta (m11))
```

```
##
## Call:
## lm(formula = salact ~ saldeb + catemp, data = emp)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -28342  -3779   -901   2601  49848
##
## Coefficients:
##                    Estimate Standardized Std. Error t value Pr(>|t|)
## (Intercept)       8.120e+03    0.000e+00  1.062e+03   7.645 1.19e-13 ***
## saldeb            1.399e+00    6.448e-01  7.003e-02  19.976  < 2e-16 ***
## catempCadre       1.727e+03    2.347e-02  1.491e+03   1.158    0.247
## catempResponsable 1.353e+04    3.029e-01  1.449e+03   9.340  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 7469 on 470 degrees of freedom
## Multiple R-squared:  0.8099, Adjusted R-squared:  0.8087
## F-statistic: 667.5 on 3 and 470 DF,  p-value: < 2.2e-16
```

```
par (mfrow = c(1,2))
visreg(m11)
```

[7] La question des interactions dans la régression sera traitée dans la section suivante.



Dans les résultats, on trouve maintenant 4 coefficients :

- l'intercept : salaire moyen de la catégorie "Secrétariat" lorsque le salaire d'embauche est égal à zéro
- le coefficient associé à "saldeb" : effet du salaire d'embauche à catégorie d'emploi constante
- le coefficient associé à "catempCadre" : différence de salaire moyen entre "Cadre" et "Secrétariat" à salaire d'embauche constant
- le coefficient associé à "catempResponsable" : différence de salaire moyen entre "Cadre" et "Responsable" à salaire d'embauche constant

L'introduction dans la régression de variables factices fait que l'on n'a pas l'effet global de la catégorie d'emploi. Autrement dit, on a l'effet de la catégorie "Cadre" par rapport à la catégorie "Secrétariat" et l'effet de la catégorie "Responsable" par rapport à la catégorie "Secrétariat", mais pas l'effet d'ensemble de la variable "catemp". Pour l'obtenir, il faut comparer deux modèles, celui comportant uniquement la variable "saldeb" (modèle `m1`) et celui comportant la variable "saldeb" et la variable "catemp" (modèle `m11`). Cette comparaison de modèles s'effectue avec la fonction `anova` :

```
anova (m1, m11)
```

```
## Analysis of Variance Table
##
## Model 1: salact ~ saldeb
## Model 2: salact ~ saldeb + catemp
##   Res.Df        RSS Df  Sum of Sq      F    Pr(>F)
## 1    472 3.1085e+10
## 2    470 2.6218e+10  2 4867716626 43.631 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Les résultats montrent un effet global significatif de la catégorie d'emploi : $F(2,470) = 43.631$, $p < 0.001$.

Dans l'exemple précédent, on a comparé un modèle avec une variable indépendante déjà présente avec un second modèle où on ajoute une variable polytomique. Pour savoir si une variable nominale polytomique a globalement un effet significatif et que l'on ne dispose pas d'un modèle avec une ou plusieurs autres variables déjà introduites, il suffit d'utiliser comme comparatif le modèle "nul", c'est-à-dire un modèle dans lequel il n'y a que la constante (l'intercept) et aucune variable indépendante. Ce modèle s'écrit : `lm (salact ~ 1,`

`data = emp`)[8]. La commande suivante compare le modèle nul avec le modèle où la catégorie d'emploi est la seule variable indépendante :

```
anova (lm(salact ~ 1, data = emp), lm(salact ~ catemp, data = emp))
```

```
## Analysis of Variance Table
##
## Model 1: salact ~ 1
## Model 2: salact ~ catemp
##   Res.Df        RSS Df  Sum of Sq      F    Pr(>F)
## 1    473 1.3792e+11
## 2    471 4.8478e+10  2 8.9438e+10 434.48 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

12.5 Les interactions dans la régression

L'écriture des modèles linéaires utilise quelques codes importants qu'il est facile à retenir :

- `y ~ x1 + x2` correspond à un modèle avec seulement les effets principaux de x1 et de x2 (signe `+`)
- `y ~ x1 + x2 + x1:x2` correspond à un modèle avec les effets principaux de x1 et de x2 et l'interaction entre les deux (signe `:`)
- `y ~ x1 * x2` correspond à un modèle avec tous les effets (signe `*`) : effets principaux et interaction (ce qui est équivalent à la spécification précédente donc).

L'interprétation et la visualisation des résultats de modèles avec interaction dépend du type des variables indépendantes : nominales ou quantitatives.

12.5.1 Interaction entre une variable nominale et une variable quantitative

Le modèle qui va être estimé ici introduit deux variables indépendantes (la catégorie d'emploi : variable nominale polytomique et le niveau d'éducation, variable quantitative) et leur interaction. Ce modèle s'écrit `lm (salact ~ educ + catemp + educ:catemp)` ou, ce qui est équivalent, `lm (salact ~ educ * catemp)`.

L'estimation de ce modèle pose de sérieux problèmes de multicolinéarité, comme le montrent les résultats suivants des coefficients *VIF* :

```
vif (lm (salact ~ educ + catemp + educ:catemp, data = emp))
```

```
##                  educ            catempCadre      catempResponsable
##              1.998558              23.777497             102.145220
##       educ:catempCadre educ:catempResponsable
##             23.029904             106.872781
```

Lorsque l'interaction comporte une ou plusieurs variables quantitatives, un moyen de réduire la multicolinéarité consiste à centrer les variables quantitatives. Autrement dit, il s'agit tout simplement de créer une nouvelle variable égale à la variable originale moins sa moyenne. Cette nouvelle variable ("educc") a alors bien sûr une moyenne nulle, mais garde la même variance :

```
emp$educc <- emp$educ - mean (emp$educ)
describe (emp[, c("educ", "educc")])
```

[8]Dans la formule, le terme `1` désigne la constante. Quand un modèle contient une variable indépendante, la constance est automatiquement incluse ce qui fait que l'écriture `y ~ x` est équivalente à `y ~ 1 + x`.

```
##       vars   n  mean   sd median trimmed  mad   min   max range  skew
## educ     1 474 13.49 2.88  12.00   13.54 4.45  8.00 21.00    13 -0.11
## educc    2 474  0.00 2.88  -1.49    0.05 4.45 -5.49  7.51    13 -0.11
##       kurtosis   se
## educ     -0.29 0.13
## educc    -0.29 0.13
```

Les indicateurs de multicolinéarité du modèle où la variable "educ" est remplacée par la variable centrée "educc" sont beaucoup plus raisonnables :

```
vif (lm (salact ~ educc + catemp + educc:catemp, data = emp))
```

```
##                   educc             catempCadre       catempResponsable
##                1.998558                3.191594                5.551951
##       educc:catempCadre educc:catempResponsable
##                3.379728                6.130321
```

Pour savoir si l'interaction a un effet significatif, nous allons commencer par comparer deux modèles ; le modèle avec seulement les effets principaux (`m12`) et le modèle avec les effets principaux et l'interaction (`m13`). Les résultats montrent que l'effet de l'interaction est statistiquement significatif :

```
m12 <- lm (salact ~ educc + catemp, data = emp)
m13 <- lm (salact ~ educc + catemp + educc:catemp, data = emp)
anova (m12,m13)
```

```
## Analysis of Variance Table
##
## Model 1: salact ~ educc + catemp
## Model 2: salact ~ educc + catemp + educc:catemp
##   Res.Df        RSS Df  Sum of Sq      F    Pr(>F)
## 1    470 4.0639e+10
## 2    468 3.8588e+10  2 2051563311 12.441 5.441e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

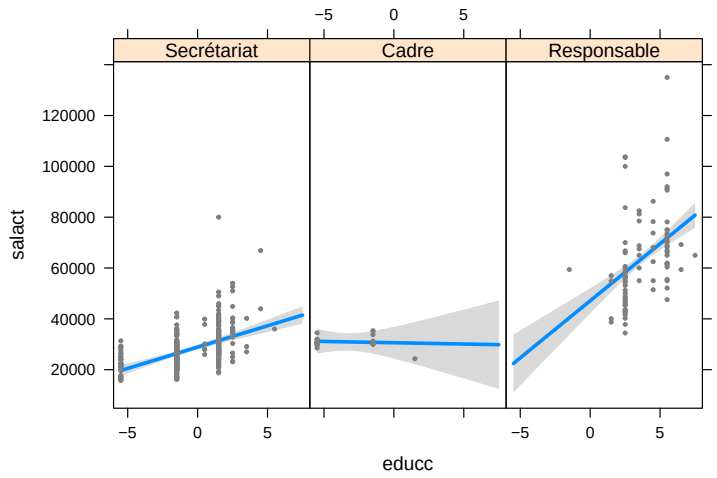
Avant de voir comment donner un sens aux différents coefficients de régression, il est toujours utilise, en présence d'interaction, d'en faire une représentation graphique. La fonction `visreg` permet de le faire facilement. Contrairement aux analyses précédentes où la fonction `visreg` a été utilisée sans arguments, plusieurs sont utiles ici pour obtenir un graphique le plus lisible possible :

- après le nom du modèle (`m13`), le nom de la variable "educc" indique la variable qui sera positionnée sur l'axe des abscisses
- l'argument `by` permet d'indiquer la seconde variable du terme de l'interaction. C'est une variable qualitative ici et on aura donctrois graphiques différents : un graphique pour la catégorie "secrétariat", un pour "cadre" et un pour "responsable"
- l'argument `layout` indique la manière de présenter les différents graphique (ici, trois graphiques côte-à-côte)[9]

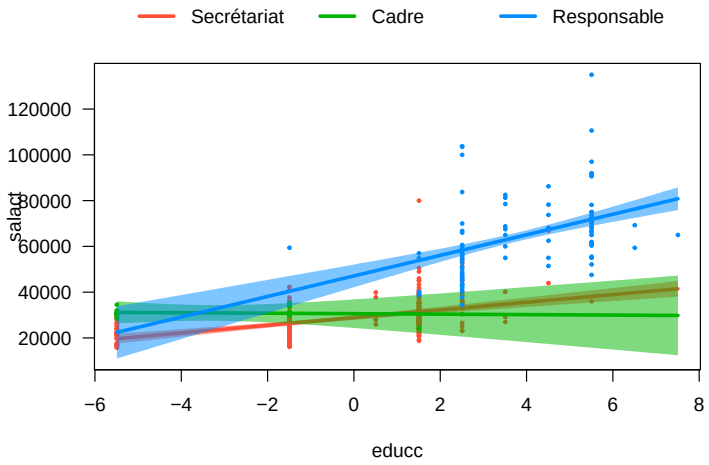
La seconde commande ci-dessous permet de superposer les trois droites de régression dans un même graphique

[9] En écrivant `layout = c(1,3)`, on aurait obtenu trois graphiques les uns en dessous des autres.

```
visreg(m13, "educc", by = "catemp", layout = c(3,1))
```




```
visreg(m13, "educc", by = "catemp", overlay = T)
```



Ce graphique montre assez clairement la nature de l'interaction, à savoir que l'effet du niveau d'éducation n'est pas le même selon la catégorie d'emploi :

- chez les responsables, l'augmentation du niveau d'études a pour conséquence une augmentation importante du salaire actuel ;
- chez les secrétaires, l'augmentation du niveau d'études a pour conséquence une augmentation beaucoup plus modérée du salaire actuel ;
- chez les cadres, il ne semble pas avoir de lien entre le niveau d'études et le salaire actuel[10].

Les différents coefficients de régression sont présentés ci-dessous. Noter que l'on n'a pas demandé les coefficients standardisés comme dans comme dans les précédentes analyses (avec la fonction `lm.beta`), car en présence d'une interaction, cette fonction ne fournit pas les coefficients corrects.

```
summary (m13)
```

[10]Il est toutefois important de noter que l'intervalle de confiance de la droite de régression chez les cadres est très étendu. Cela est dû au fait qu'il y a peu de cadres avec des niveaux d'éducation élevés, ce qui rend les estimations du côté droit du graphique très peu certaines.

```
##
## Call:
## lm(formula = salact ~ educc + catemp + educc:catemp, data = emp)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -24287  -4333   -838   2824  63163
##
## Coefficients:
##                          Estimate Std. Error t value Pr(>|t|)
## (Intercept)               28884.5      493.4  58.543  < 2e-16 ***
## educc                      1676.8      204.6   8.196 2.41e-15 ***
## catempCadre                1716.3     3214.8   0.534   0.5937
## catempResponsable         18215.0     2573.6   7.078 5.39e-12 ***
## educc:catempCadre         -1779.0      828.0  -2.149   0.0322 *
## educc:catempResponsable    2814.0      651.2   4.321 1.90e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 9080 on 468 degrees of freedom
## Multiple R-squared:  0.7202, Adjusted R-squared:  0.7172
## F-statistic: 240.9 on 5 and 468 DF,  p-value: < 2.2e-16
```

Ces coefficients s'interprètent de la manière suivante :

- l'intercept (28884.5), c'est le salaire actuel moyen des secrétaires ayant le niveau d'éducation moyen de l'ensemble des salariés (i.e., le niveau d'éducation centrée égal à zéro, correspond à la moyenne du niveau d'éducation) ;
- le coefficient de la variable "educc" indique l'augmentation moyenne du salaire (+1676.8) pour une année d'éducation supplémentaire chez les secrétaires ;
- le coefficient de la variable factice "catempCadre" correspond à la différence de salaire entre les cadres et les secrétaires (+1716.3) pour un niveau d'éducation moyen (educc = 0) (noter que cette différence n'est pas statistiquement significative) ;
- le coefficient de la variable factice "catempResponsable" correspond à la différence de salaire entre les responsables et les secrétaires (+18215.0) pour un niveau d'éducation moyen (educc = 0) ;
- le premier coefficient de l'interaction (educc:catempCadre) est l'écart entre les pentes des droites de régression chez les secrétaires et chez les cadres. Autrement dit, chez les secrétaires, une année d'éducation supplémentaire entraîne en moyenne un salaire supérieur de 1676.8 ; chez les cadres, une année d'éducation supplémentaire entraine une variation de salaire de -102.2 (1676.8-1779.0) ;
- de la même manière, le second coefficient de l'interaction (educc:catempResponsable) est l'écart entre les pentes des droites de régression des secrétaires et des responsables. Une année d'étude en plus chez les secrétaires augmente le salaire en moyenne de 1676.8 ; chez les responsables, une année supplémentaire augmente le salaire de 1676.8 + 2814.0 = 4490.8.

12.5.2 Interaction entre deux variables indépendantes quantitatives

Le salaire d'embauche ("saldeb") et l'expérience dans la fonction ("exp") seront les deux variables quantitatives choisies pour illustrer ce type d'interaction dans le modèle linéaire. Comme précédemment, l'introduction de ces deux variables dans un modèle créée des problèmes (modérés) de multicolinéarité :

```
vif (lm (salact ~ saldeb * exp, data = emp))
```

```
##     saldeb        exp saldeb:exp
##   3.138579   7.168528   9.602386
```

L'utilisation de variables centrées permet de faire disparaître ces problèmes :

```
emp$saldebc <- emp$saldeb - mean (emp$saldeb)
emp$expc <- emp$exp - mean (emp$exp)
vif (lm (salact ~ saldebc * expc, data = emp))
```

```
##      saldebc         expc saldebc:expc
##     1.092769     1.050974     1.134841
```

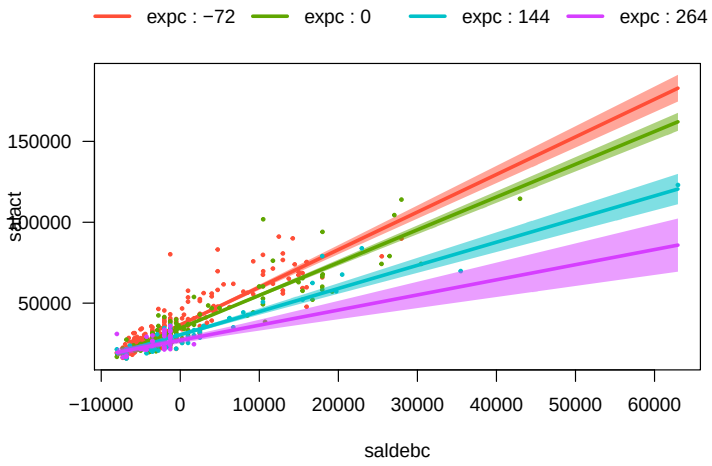
L'interaction entre deux variables quantitatives ajoute un seul effet dans le modèle (il n'y a pas de variables factices) et il n'est donc pas nécessaire de comparer un modèle sans et avec interaction pour savoir si celle-ci est significative. Le coefficient du terme de l'interaction répond directement à cette question et celui-ci est bien significatif comme le montrent les résultats ci-dessous :

```
m14 <- lm (salact ~ saldebc * expc, data = emp)
summary (m14)
```

```
##
## Call:
## lm(formula = salact ~ saldebc * expc, data = emp)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -25951  -3767  -1031   2752  46554
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)   3.456e+04  3.354e+02 103.050  < 2e-16 ***
## saldebc       2.026e+00  4.453e-02  45.487  < 2e-16 ***
## expc         -2.848e+01  3.303e+00  -8.622  < 2e-16 ***
## saldebc:expc -4.132e-03  5.122e-04  -8.067 6.05e-15 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 7292 on 470 degrees of freedom
## Multiple R-squared:  0.8188, Adjusted R-squared:  0.8177
## F-statistic:   708 on 3 and 470 DF,  p-value: < 2.2e-16
```

La visualisation de l'effet interactif par la fonction `visreg` nécessite de définir des niveaux à l'une des variable quantitative pour représenter les droites de régression de l'autre variable à chacun de ces niveaux. Et il faut décider laquelle des variables quantitatives sera sur l'axe des abscisses. Dans l'illustration suivante, c'est le salaire d'embauche qui a été choisi pour figure en abscisse :

```
visreg (m14, "saldebc", by = "expc", breaks = c(-72, 0, 144, 264), overlay = T)
```

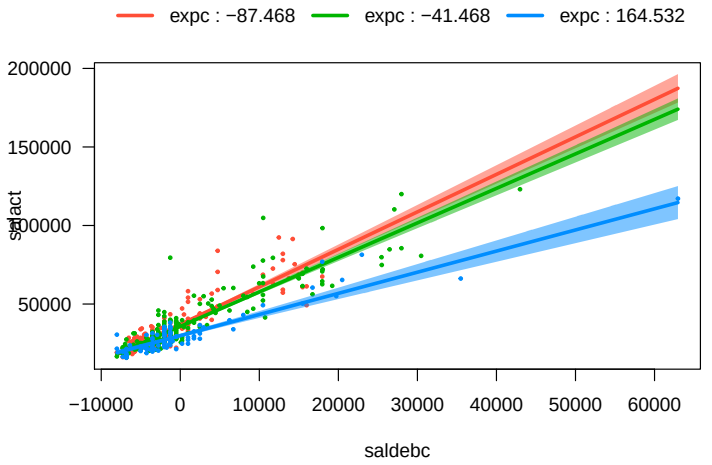


Avant d'expliciter à quoi correspond l'argument `breaks`, notons tout d'abord que ce graphique montre clairement que l'effet du salaire d'embauche sur le salaire actuel est d'autant plus important que les personnes ont une faible expérience dans le poste (un résultat assez trivial somme toute !). Autrement dit, l'influence du salaire d'embauche sur le salaire actuel est très élevé chez les personnes ayant peu d'expérience et faible chez les personnes ayant beaucoup d'expériences.

Dans ce graphique la droite violette correspond à des personnes ayant une trentaine d'années d'expériences, la bleue à une vingtaine d'année, la verte à huit ans environ et la rouge à deux ans environ. Comment ont été déterminées ces valeurs ? La variable expérience introduite dans la régression est une variable centrée. La valeur 0 de la variable centrée correspond donc à la moyenne de la variable originale (96.5 mois, soit 8 années environ). Une valeur de "-72" sur la variable centrée, c'est 6 (72 = 6 * 12 mois) années de moins que l'expérience moyenne, soit 2 ans environ. Une valeur de "+144" correspond à 12 ans de plus que la moyenne (soit 20 ans environ) et une valeur de "+264" à 22 ans de plus (soit 30 ans environ).

Ce sont donc ces valeurs (-60, 0, 100, 200) qui ont été indiquées comme paramètres dans l'argument `breaks` de la fonction `visreg`. Si cet argument n'est pas spécifié, trois niveaux sont définis par défaut correspondant respectivements aux 10^{e}, 50^{e} et 90^{e} déciles de la variable :

```
visreg (m14, "saldebc", by = "expc", overlay = T)
```



Enfin, voyons l'interprétation des coefficients de régression :

```
summary (m14)
```

```
##
## Call:
## lm(formula = salact ~ saldebc * expc, data = emp)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -25951  -3767  -1031   2752  46554
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)   3.456e+04  3.354e+02 103.050  < 2e-16 ***
## saldebc       2.026e+00  4.453e-02  45.487  < 2e-16 ***
## expc         -2.848e+01  3.303e+00  -8.622  < 2e-16 ***
## saldebc:expc -4.132e-03  5.122e-04  -8.067 6.05e-15 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 7292 on 470 degrees of freedom
## Multiple R-squared:  0.8188, Adjusted R-squared:  0.8177
## F-statistic:   708 on 3 and 470 DF,  p-value: < 2.2e-16
```

- L'intercept (34560) est le salaire actuel moyen estimé des personnes ayant un salaire d'embauche moyen (saldebc = 0) et une expérience moyenne (expc = 0).
- Le coefficient de régression de "saldebc" indique qu'une différence de 1 euro du salaire lors de l'embauche entraine une différence moyenne de 2.026 euros entre les salaires actuels (pour une expérience moyenne)
- Le coefficient de régression de "expc" indique que les personnes qui ont un mois supplémentaire d'expérience ont en moyenne un salaire actuel inférieur de 28.48 euros (pour un salaire d'embauche moyen)
- Le coefficient de l'interaction (saldebc:expc) permet de quantifier la diminution de l'**effet** du salaire pour chaque incrément de l'expérience. La visualisation de la régression pour différents niveau de l'expérience permet de se représenter plus facilement à quoi correspond ce coefficient.

12.5.3 Interaction entre deux variables indépendantes qualitatives

Le sexe et la catégorie d'emploi sont deux variables qualitatives du fichier "emp". Leur croisement indique qu'il n'y a aucune femme cadre. Il n'est donc pas possible d'introduire dans un modèle de régression l'interaction entre ces deux variables.

```
table (emp$catemp, emp$sexe)
```

```
##
##               Féminin Masculin
##   Secrétariat     206      157
##   Cadre             0       27
##   Responsable      10       74
```

La minorité (comprendre l'appartenance à une minorité ethnique) est une autre variable qualitative dichotomique. Son croisement avec la catégorie d'emploi fournit les résultats suivants :

```
table (emp$catemp, emp$minorite)
```

```
##
##               Non Oui
```

```
##   Secrétariat 276  87
##   Cadre        14  13
##   Responsable  80   4
```

Il n'y a que quatre responsables appartenant à une minorité ethnique. Ce faible effectif devrait nous interdire de tester un modèle avec l'interaction entre ces deux variables. Toutefois, uniquement afin de permettre d'exposer la manière de tester et d'interpréter une interaction entre deux variables qualitatives, nous utiliserons ces deux variables en gardant à l'esprit que dans une telle situation, cette analyse ne devrait pas être menée.

Le premier modèle estimé (`m15`) est un modèle comprenant uniquement les effets principaux de la minorité et de la catégorie d'emploi. Dans le second (`m16`), on ajoute l'interaction entre les deux. La comparaison entre ces deux modèles montre que l'effet de l'interaction est statistiquement significatif : $F(2,468) = 3.88$, $p = 0.02$.

```
m15 <- lm (salact ~ minorite + catemp, data = emp)
m16 <- lm (salact ~ minorite + catemp + minorite:catemp, data = emp)
anova (m15,m16)
```

```
## Analysis of Variance Table
##
## Model 1: salact ~ minorite + catemp
## Model 2: salact ~ minorite + catemp + minorite:catemp
##   Res.Df        RSS Df Sum of Sq      F  Pr(>F)
## 1    470 4.8363e+10
## 2    468 4.7575e+10  2 788578413 3.8787 0.02135 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

On notera au passage que l'analyse de variance fournit les mêmes résultats quant à l'effet de l'interaction[11] :

```
ezANOVA(data = emp, wid = sujet, dv = salact, between = .list (catemp, minorite), detailed = T)
```

```
## Warning: Converting "sujet" to factor for ANOVA.
```

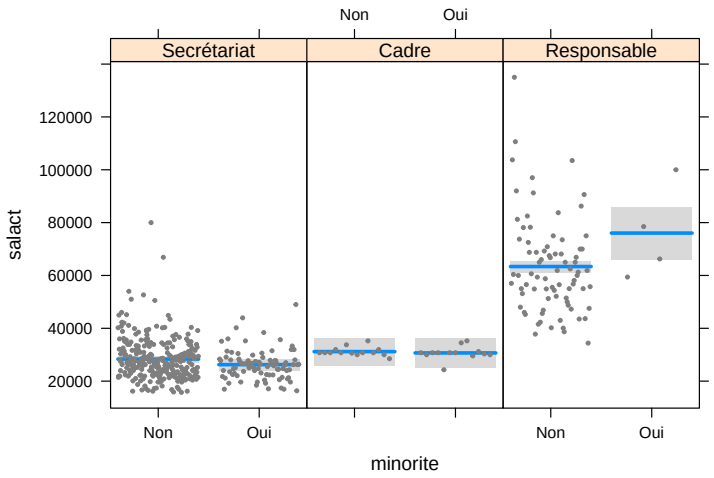
```
## Warning: Data is unbalanced (unequal N per group). Make sure you specified
## a well-considered value for the type argument to ezANOVA().
```

```
## $ANOVA
##            Effect DFn DFd         SSn         SSd          F             p
## 1          catemp   2 468 85215976229 47574671011 419.141909 4.838866e-105
## 2        minorite   1 468   114762086 47574671011   1.128934  2.885508e-01
## 3 catemp:minorite   2 468   788578413 47574671011   3.878689  2.134593e-02
##   p<.05         ges
## 1     * 0.641731763
## 2       0.002406447
## 3     * 0.016305323
##
## $`Levene's Test for Homogeneity of Variance`
##   DFn DFd        SSn         SSd        F            p p<.05
## 1   5 468 5355781325 23567472554 21.27089 3.440109e-19     *
```

L'interprétation de l'interaction est facilitée par l'examen du graphique des moyennes. La fonction `visreg` permet d'obtenir le graphique suivant :

[11] Lorsque les effectifs des groupes issus du croisement entre deux variables qualitatives ne sont pas les mêmes (comme c'est le cas ici), les estimations d'un modèle linéaire donnent des résultats différents selon la manière dont on calcule les différentes sommes de carrés. Il s'agit d'une question techniquement complexe dont il ne sera pas fait état dans ce chapitre. Mais, en l'occurrence, la fonction `lm` et `ezANOVA` utilisent, par défaut, le même calcul de la somme de carrés, dite de type 2.

```
par (mfrow = c(1,1))
visreg(m16, "minorite", by = "catemp", layout = c(3,1))
```



Ce graphique montre visuellement que l'effet de la variable "minorite" est quasi inexistant dans les catégories "secrétariat" et "cadre". Il y a en revanche une différence importante de salaire moyen due à la minorité dans la catégorie "reponsable"[12].

```
summary (m16)
```

```
##
## Call:
## lm(formula = salact ~ minorite + catemp + minorite:catemp, data = emp)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -28965  -5541   -891   3469  71625
##
## Coefficients:
##                               Estimate Std. Error t value Pr(>|t|)
## (Intercept)                    28341.1      606.9  46.699  < 2e-16 ***
## minoriteOui                    -2096.8     1239.7  -1.691  0.09142 .
## catempCadre                     2837.5     2762.1   1.027  0.30482
## catempResponsable              35033.7     1280.2  27.365  < 2e-16 ***
## minoriteOui:catempCadre         1599.0     4076.5   0.392  0.69504
## minoriteOui:catempResponsable  14759.5     5312.4   2.778  0.00568 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 10080 on 468 degrees of freedom
## Multiple R-squared:  0.655,  Adjusted R-squared:  0.6514
## F-statistic: 177.7 on 5 and 468 DF,  p-value: < 2.2e-16
```

L'intercept correspond toujours au salaire actuel moyen estimé lorsque toutes les variables indépendantes sont égales à zéro, autrement dit, ici, au salaire moyen estimé des secrétaires n'appartenant pas à une minorité ethnique.

[12] On se rappellera toutefois (et cela est visible sur le graphique) que le salaire moyen des responsables issus d'une minorité n'est calculé que sur quatre individus, ce qui rend cette estimation très peu fiable.

Le coefficient de régression de la variable factice “minoriteOui” est la différence entre le salaire moyen des personnes issues d'une minorité ethnique par rapport aux personnes qui ne sont pas issues d'une minorité ethnique, la catégorie d'emploi étant contrôlée.

Le coefficient de régression de la variable factice “catempCadre” est la différence entre le salaire moyen des cadres par rapport aux secrétaires, la variable “minorite” étant contrôlée.

Le coefficient de régression de la variable factice “catempResponsable” est la différence entre le salaire moyen des responsables par rapport aux secrétaires, la variable “minorite” étant contrôlée.

L'interprétation des coefficients de régression de l'interaction nécessite un peu de gymnastique arithmétique qui passe tout d'abord par l'écriture de l'équation de régression :

$$salact = 28341.10 - (2096.8 \times minorite) + (2837.5 \times cadre) + (35033.7 \times responsable) + (1599.0 \times minorite \times cadre) + (14759.5 \times mi$$

Avec cette équation, on peut calculer les moyennes estimées des différents groupes.

- les secrétaires ne faisant pas partie d'une minorité codent “0” sur la variable “minorite”, “0” sur la variable “cadre” et “0” sur la variable “responsable”. L'équation devient donc :

$$\begin{aligned} salact &= 28341.10 - (2096.8 \times 0) + (2837.5 \times 0) + (35033.7 \times 0) + \\ &(1599.0 \times 0 \times 0) + (14759.5 \times 0 \times 0) \\ &= 28341.10 \end{aligned}$$

Et on retrouve bien la valeur de l'intercept.

- les secrétaires faisant partie d'une minorité codent “1” sur la variable “minorite”, “0” sur la variable “cadre” et “0” sur la variable “responsable”. L'équation devient :

$$salact = 28341.10 - (2096.8 \times 1) + (2837.5 \times 0) + (35033.7 \times 0) + (1599.0 \times 1 \times 0) + (14759.5 \times 1 \times 0) = 26244.25$$

- les cadres ne faisant pas partie d'une minorité codent “0” sur la variable “minorite”, “1” sur la variable “cadre” et “0” sur la variable “responsable” :

$$salact = 28341.10 - (2096.8 \times 0) + (2837.5 \times 1) + (35033.7 \times 0) + (1599.0 \times 0 \times 1) + (14759.5 \times 0 \times 0) = 31178.57$$

- les cadres faisant partie d'une minorité codent “1” sur la variable “minorite”, “1” sur la variable “cadre” et “0” sur la variable “responsable” :

$$salact = 28341.10 - (2096.8 \times 1) + (2837.5 \times 1) + (35033.7 \times 0) + (1599.0 \times 1 \times 1) + (14759.5 \times 1 \times 0) = 30680.77$$

- les responsables ne faisant pas partie d'une minorité codent “0” sur la variable “minorite”, “0” sur la variable “cadre” et “1” sur la variable “responsable” :

$$salact = 28341.10 - (2096.8 \times 0) + (2837.5 \times 0) + (35033.7 \times 1) + (1599.0 \times 0 \times 0) + (14759.5 \times 0 \times 1) = 63374.81$$

- les responsables faisant partie d'une minorité codent “1” sur la variable “minorite”, “0” sur la variable “cadre” et “1” sur la variable “responsable” :

$$salact = 28341.10 - (2096.8 \times 1) + (2837.5 \times 0) + (35033.7 \times 1) + (1599.0 \times 1 \times 0) + (14759.5 \times 1 \times 1) = 76037.5$$

Les différentes moyennes estimées sont résumées dans le tableau ci-dessous et ce sont ces moyennes que l'on retrouve dans le graphique que l'on a vu auparavant :

	minorité: non	minorité : oui
Secrétaires	28341.09	26244.25
Cadres	31178.57	30680.77
Responsables	63374.81	76037.50

Le premier coefficient qui teste l'interaction vaut 1599.0 ("minoriteOui:catempCadre"). Il correspond à la différence entre deux différences :

- la moyenne des cadres faisant partie d'une minorité (30680.77) et celle des secrétaires faisant partie d'une minorité (26244.25) ;
- la moyenne des cadres ne faisant pas partie d'une minorité (31178.57) et celle des secrétaires ne faisant pas partie d'une minorité (28341.09).

Soit : (30680.77 - 26244.25) - (31178.57 - 28341.09) = 1599.0

Plus simplement peut-être, ce coefficient indique si l'écart entre cadres et secrétaires est plus ou moins important pour les personnes faisant partie d'une minorité que celles qui n'en font pas partie (i.e. l'effet de la minorité est-il le même quand on compare cadres et secrétaires). Ce coefficient n'est pas significatif, donc l'écart entre ces deux catégories d'emploi ne dépend pas du fait d'appartenir à une minorité.

Selon la même logique, le second coefficient qui teste l'interaction vaut 14759.5 et il est statistiquement significatif. Cela signifie que l'écart entre les salaires des responsables par rapport à celui des secrétaires est significativement plus important pour les personnes faisant partie d'une minorité que pour les personnes n'en faisant pas partie.

Bibliographie

Bates, D., Mächler, M., Bolker, B., & Walker, S. (2015). Fitting linear mixed-effects models using lme4. *Journal of Statistical Software*, *67*(1), 1–48. https://doi.org/10.18637/jss.v067.i01

Behrendt, S. (2014). Lm.Beta: Add standardized regression coefficients to lm-objects [R package version 1.5-1]. https://CRAN.R-project.org/package=lm.beta.

Breheny, P., & Burchett, W. (2017). Visreg: Visualization of regression models (R package version 2.4-1). https://CRAN.R-project.org/package=visreg.

Chang, W., & Wickham, H. (2016). Ggvis: Interactive grammar of graphics [R package version 0.4.3]. https://CRAN.R-project.org/package=ggvis.

Chang, W., Cheng, J., Allaire, J., Xie, Y., & McPherson, J. (2017). Shiny: Web application framework for R [R package version 1.0.5]. https://CRAN.R-project.org/package=shiny.

Dragulescu, A. A. (2014). Xlsx: Read, write, format Excel 2007 and Excel 97/2000/XP/2003 files [R package version 0.5.7]. https://CRAN.R-project.org/package=xlsx.

Faraway, J. (2016). Faraway: Functions and datasets for books by Julian Faraway [R package version 1.0.7]. https://CRAN.R-project.org/package=faraway.

Fellows, I. (2012). DeduceR: A data analysis GUI for R. *Journal of Statistical Software*, *49*(8), 1–15.

Fox, J. (2017). *Using the R commander: A point-and-click interface for R*. Boca Raton FL: Chapman and Hall/CRC Press.

Fox, J., & Weisberg, S. (2011). *An R companion to applied regression* (second edition). Thousand Oaks, CA: Sage.

Helbig, M., Theus, M., & Urbanek, S. (2005). JGR: Java GUI for R. *Statistical Computing and Graphics*, *16*(2), 9–11.

Holzinger, K. J., & Swineford, F. (1937). *A study in factor analysis: The stability of a bifactor solution* (Supplementary Educational Monograph, no. 48). Chicago: University of Chicago Press.

Hothorn, T., Bretz, F., & Westfall, P. (2008). Simultaneous inference in general parametric models. *Biometrical Journal*, *50*(3), 346–363.

Hugh-Jones, D. (2017). Anim.Plots: Simple animated plots for R [R package version 0.2]. https://CRAN.R-project.org/package=anim.plots.

Lawrence, M. A. (2016). Ez: Easy analysis and visualization of factorial experiments [R package version 4.4-0]. https://CRAN.R-project.org/package=ez.

Lemon, J., & Grosjean, P. (2015). Prettyr: Pretty descriptive stats [R package version 2.2]. https://CRAN.R-project.org/package=prettyR.

Michalke, M. (2014). RKWarddev: A collection of tools for RKWard plugin development (Version 0.06-5).

http://rkward.sourceforge.net.

Mirai Solutions GmbH. (2017). XLConnect: Excel connector for R [R package version 0.2-13]. https://CRAN.R-project.org/package=XLConnect.

Pedersen, T. L. (2016). Tweenr: Interpolate data for smooth animations [R package version 0.1.5]. https://CRAN.R-project.org/package=tweenr.

R Core Team. (2017a). Foreign: Read data stored by 'Minitab', 'S', 'SAS', 'SPSS', 'Stata', 'Systat', 'Weka', 'dBase', . [R package version 0.8-69]. https://CRAN.R-project.org/package=foreign.

R Core Team. (2017b). *R: A language and environment for statistical computing*. Vienna, Austria: R Foundation for Statistical Computing.

Revelle, W. (2017). Psych: Procedures for psychological, psychometric, and personality research [R package version 1.7.5]. https://CRAN.R-project.org/package=psych.

Sievert, C., Parmer, C., Hocking, T., Chamberlain, S., Ram, K., Corvellec, M., & Despouy, P. (2017). Plotly: Create interactive web graphics via 'plotly.Js' [R package version 4.7.1]. https://CRAN.R-project.org/package=plotly.

Warnes, G. R., Bolker, B., Gorjanc, G., Grothendieck, G., Korosec, A., Lumley, T., ... others. (2017). Gdata: Various R programming tools for data manipulation [R package version 2.18.0]. https://CRAN.R-project.org/package=gdata.

Warnes, G. R., Bolker, B., Lumley, T., & Johnson, R. C. (2015). Gmodels: Various R programming tools for model fitting [R package version 2.16.2]. https://CRAN.R-project.org/package=gmodels.

Wickham, H. (2009). *Ggplot2: Elegant graphics for data analysis*. New York: Springer-Verlag.

Wickham, H. (2017). Tidyr: Easily tidy data with 'spread()' and 'gather()' functions [R package version 0.6.3]. https://CRAN.R-project.org/package=tidyr.

Wickham, H., & Grolemund, G. (2016). *R for data science: Import, tidy, transform, visualize, and model data* (First edition). Sebastopol, CA: O'Reilly Media.

Williams, G. J. (2011). *Data mining with rattle and R: The art of excavating data for knowledge discovery*. New York, NY: Springer.

Xie, Y. (2013). Animation: An R package for creating animations and demonstrating statistical methods. *Journal of Statistical Software*, *53*(1), 1–27.

Xie, Y. (2014). Knitr: A comprehensive tool for reproducible research in r. In V. Stodden, F. Leisch, & R. D. Peng (Eds.), *Implementing reproducible computational research*. Boca Raton, Florida: Chapman and Hall/CRC.

Xie, Y. (2016). *Bookdown: Authoring books and technical documents with r markdown*. Boca Raton, Florida: Chapman and Hall/CRC.

Zeileis, A., Meyer, D., & Hornik, K. (2007). Residual-based shadings for visualizing (conditional) independence. *Journal of Computational and Graphical Statistics*, *16*(3), 507–525.