Gestion des données avec R

Christophe Lalanne & Bruno Falissard

Table des matières

1 Introduction

La gestion des données sous R n'est pas aussi évidente qu'il n'y paraît au premier abord. C'est essentiellement dû au fait que l'on ne "voit pas" les données comme sur un tableur de type Excel. Toutefois, R offre des outils puissants de recodage des variables et de reformatage des tableaux de données, et permet de lire quasiment tous les formats de fichiers de données utilisés dans le domaine statistique.

2 Importation de fichiers CSV

2.1 Structure du fichier de données

Considérons le fichier de données `smp2.csv` qui regroupe les données sur l'étude de santé mentale en prison. Ce fichier comporte 26 variables et 799 observations (individus ou unités statistiques). Il s'agit d'un fichier de type CSV (comma separated values) que l'on peut ouvrir avec un tableur de type Excel ou n'importe quel éditeur de texte. Souvent d'ailleurs, lorsque l'on double-clique sur un fichier portant cette extension (`.csv`), c'est l'application Excel (ou Open Office, par exemple) qui est proposée pour lire ce fichier. Voici à quoi ressemble ce fichier en mode texte :

Ce fichier est structuré de la manière suivante : sur la première ligne figure le nom des variables. Les données de chaque individu pour chacune de ces variables sont reportées sur une ligne séparée. Les données (nom de variable sur la première ligne, ou valeur prise par une variable pour un individu sur les lignes suivantes), appelé "champ", sont séparées par un même symbole, appelé séparateur de champ, ici un point-virgule. D'autres séparateurs de champ peuvent être utilisés, par exemple des virgules, des taquets de tabulation ou de simples espaces.

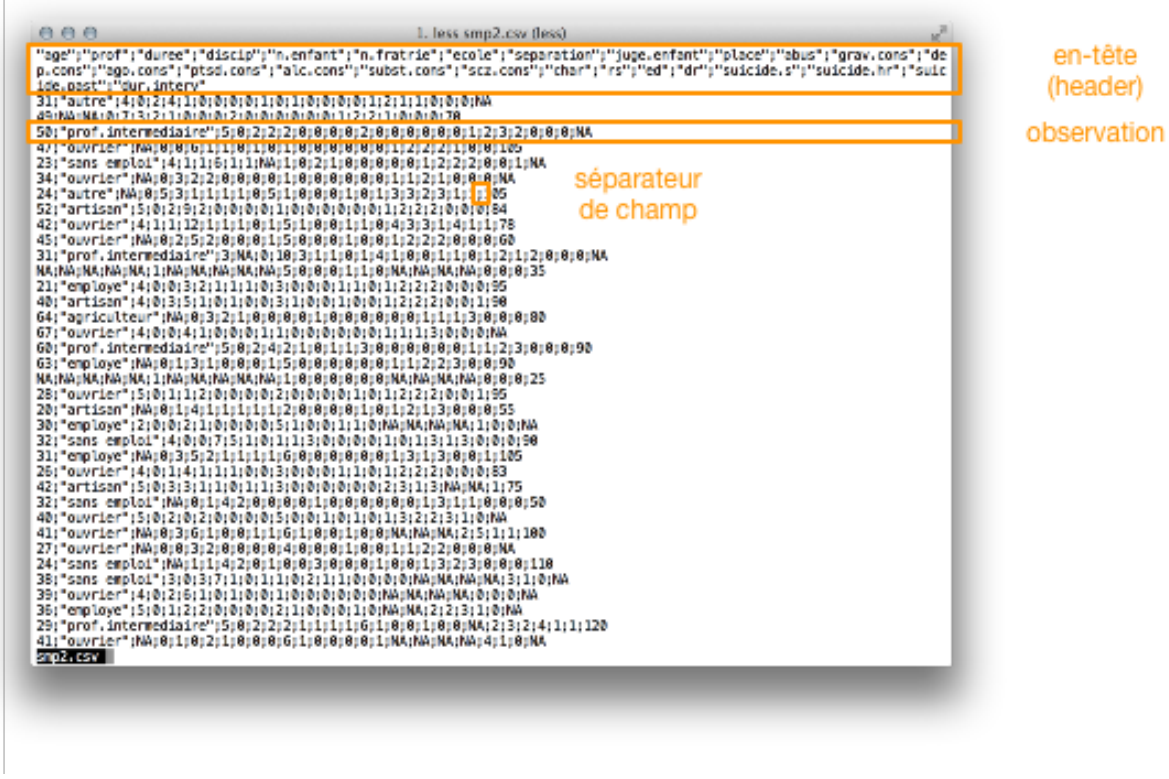

FIGURE 1 – Contenu du fichier smp2.csv en vue texte

Le séparateur décimal quant à lui permet d'indiquer à R comment sont représentés les nombres à virgules. Par défaut, R utilise la notation anglo-saxonne (le séparateur décimal est alors un point, par exemple 9.2), sauf dans le cas de la commande `read.csv2()` où l'on considère que le séparateur décimal suit la notation française (une virgule, comme dans 9,2). Évidemment, il y a des situations impossibles : utiliser comme séparateur de champs des virgules imposera le point comme séparateur décimal, autrement R n'a aucun moyen d'identifier correctement le nombre de champs présents sur chaque ligne du fichier.

Si aucune ligne d'en-tête n'est présente, il faudra préciser l'option `header = FALSE`, et éventuellement fournir le nom des variables sous forme de liste via l'option `col.names =`. Cela dit il est tout aussi simple d'utiliser la commande `colnames()` après avoir importé le fichier.

Pour importer des fichiers CSV sous R, on utilise la commande `read.csv()` ou `read.csv2()`, qui reposent en fait sur la commande `read.table()`, mais avec des options par défaut : `sep = ;/dec = ,` dans le cas de `read.csv2()`, et `sep = ,/dec = .` dans le cas de `read.csv()`. Dans tous les cas, on suppose que `header = TRUE`, c'est-à-dire que le fichier comporte bien une ligne d'en-tête.

Avant d'importer un fichier, il faut s'assurer que R connaît l'endroit où ce fichier a été enregistré. Pour cela, deux solutions : soit l'on indique le chemin d'accès complet au fichier, soit on change le répertoire de travail courant pour indiquer le répertoire dans lequel le fichier a été enregistré. Dans le premier cas, on aura donc une instruction du style :

```
smp <- read.csv2("/Users/chl/mooc/smp2.csv")
```

qui signifie que sur un Mac, le fichier `smp2.csv` se trouve dans le sous-répertoire `mooc` du répertoire `Documents` du compte utilisateur. Sous Windows, on utilisera par exemple `C:/utilisateur/chl/mooc/smp2.csv`. Dans le second cas, on utilisera soit la commande `setwd()` pour définir le répertoire de travail, soit le menu disponible dans le gestionnaire de fichiers de RStudio, comme l'illustre la figure suivante.

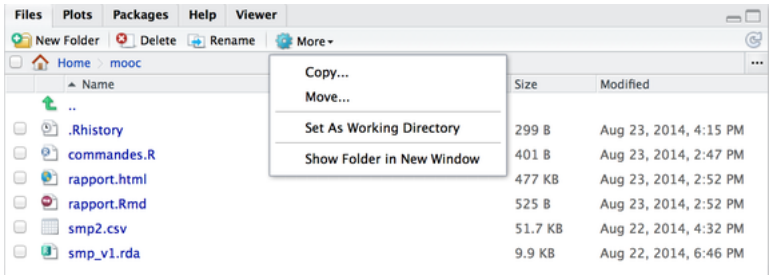

FIGURE 2 – Définir le répertoire courant comme répertoire de travail

En supposant que le répertoire de travail ait correctement été indiqué, voici comment l'on peut charger les données `smp2.csv` :

```
smp <- read.csv2("smp2.csv")
```

Concernant la structure du fichier de données une fois qu'il a été importé, il est conseillé de taper systématiquement ces 3 commandes qui permettent de vérifier la taille du tableau de données (nombre de lignes et nombre de colonnes, correspondant, respectivement, au nombre d'individus et de variables), le nom des variables (ce qui permet, entre autres, de s'assurer que la ligne d'en-tête a bien été lue correctement), et le type de représentation des variables (`int` ou `double` pour les variables numériques, `char` ou `factor` pour les variables qualitatives). Voici ce que donnent ces commandes pour le fichier `smp2.csv`.

```r
dim(smp)
```

```
## [1] 799  26
```

```r
names(smp)
```

```
##  [1] "age"          "prof"         "duree"        "discip"
##  [5] "n.enfant"     "n.fratrie"    "ecole"        "separation"
##  [9] "juge.enfant"  "place"        "abus"         "grav.cons"
## [13] "dep.cons"     "ago.cons"     "ptsd.cons"    "alc.cons"
## [17] "subst.cons"   "scz.cons"     "char"         "rs"
## [21] "ed"           "dr"           "suicide.s"    "suicide.hr"
## [25] "suicide.past" "dur.interv"
```

```r
str(smp)
```

```
## 'data.frame':    799 obs. of  26 variables:
##  $ age         : int  31 49 50 47 23 34 24 52 42 45 ...
##  $ prof        : Factor w/ 8 levels "agriculteur",..: 3 NA 7 6 8 6 3 2 6 6 ...
##  $ duree       : int  4 NA 5 NA 4 NA NA 5 4 NA ...
##  $ discip      : int  0 0 0 0 1 0 0 0 1 0 ...
##  $ n.enfant    : int  2 7 2 0 1 3 5 2 1 2 ...
##  $ n.fratrie   : int  4 3 2 6 6 2 3 9 12 5 ...
##  $ ecole       : int  1 2 2 1 1 2 1 2 1 2 ...
##  $ separation  : int  0 1 0 1 1 0 1 0 1 0 ...
##  $ juge.enfant : int  0 0 0 0 NA 0 1 0 1 0 ...
##  $ place       : int  0 0 0 1 1 0 1 0 0 0 ...
##  $ abus        : int  0 0 0 0 0 0 0 0 1 1 ...
##  $ grav.cons   : int  1 2 2 1 2 1 5 1 5 5 ...
##  $ dep.cons    : int  0 0 0 0 1 0 1 0 1 0 ...
##  $ ago.cons    : int  1 0 0 0 0 0 0 0 0 0 ...
##  $ ptsd.cons   : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ alc.cons    : int  0 0 0 0 0 0 0 0 1 1 ...
##  $ subst.cons  : int  0 0 0 0 0 0 1 0 1 0 ...
##  $ scz.cons    : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ char        : int  1 1 1 1 1 1 1 1 4 1 ...
##  $ rs          : int  2 2 2 2 2 1 3 2 3 2 ...
##  $ ed          : int  1 2 3 2 2 2 3 2 3 2 ...
##  $ dr          : int  1 1 2 2 2 1 2 2 1 2 ...
##  $ suicide.s   : int  0 0 0 1 0 0 3 0 4 0 ...
##  $ suicide.hr  : int  0 0 0 0 0 0 1 0 1 0 ...
##  $ suicide.past: int  0 0 0 0 1 0 1 0 1 0 ...
##  $ dur.interv  : int  NA 70 NA 105 NA NA 105 84 78 60 ...
```

XLS ou CSV. Il est également possible de lire directement des fichiers Excel, voire une sous-partie d'une feuille de calcul (par exemple, une zone de plage `A2:C5`, soit 12 cellules au total), à l'aide de packages

spécialisés. Cela dit, comme il est tout aussi simple d'exporter les données au format CSV depuis Excel, et que ce type de format de données pose moins de problème de compatibilité de version entre les logiciels et les systèmes d'exploitation, on préférera généralement le format CSV.

2.2 Le concept data frame

Une fois importé dans R, le fichier de données devient un tableau rectangulaire appelé "data frame" dans le jargon technique R. Ici, notre tableau de données est accessible via le data frame `smp`, qui est le nom de variable à laquelle nous avons associé les données lors de la lecture avec `read.csv2()`.

```
class(smp)
```

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

Il s'agit en fait d'une structure de données à deux dimensions (lignes = observations, colonnes = variables) contenant des données potentiellement de type mixte (nombres et chaînes de caractères). Une colonne comprendra ainsi toujours des objets du même type (par exemple des nombres pour dénoter les valeurs prises par une variable numérique), mais les différentes colonnes pourront contenir des objets de type différents (des nombres dans l'une, des caractères dans l'autre). Qui plus est, chaque ligne est identifiée par un identificateur unique, appelé "rowname".

```
rownames(smp)[1:10]
```

```
##  [1] "1"  "2"  "3"  "4"  "5"  "6"  "7"  "8"  "9"  "10"
```

Il est important de s'assurer que les données sont bien représentées comme on le souhaite, en particulier que les variables qualitatives avec un déterminé de modalités (ou niveaux) sont bien traitées comme telle par R. En l'occurence, leur type doit être `factor`. C'est le cas de la variable `prof` dans le data frame `smp`.

```
summary(smp)
```

```
##       age                     prof         duree         discip
##  Min.   :19.0   ouvrier           :227   Min.   :1.0   Min.   :0.000
##  1st Qu.:28.0   sans emploi       :222   1st Qu.:4.0   1st Qu.:0.000
##  Median :37.0   employe           :135   Median :5.0   Median :0.000
##  Mean   :38.9   artisan           : 90   Mean   :4.3   Mean   :0.232
##  3rd Qu.:48.0   prof.intermediaire: 58   3rd Qu.:5.0   3rd Qu.:0.000
##  Max.   :83.0   (Other)           : 61   Max.   :5.0   Max.   :1.000
##  NA's   :2      NA's              :  6   NA's   :223   NA's   :6
##     n.enfant       n.fratrie         ecole        separation
##  Min.   : 0.00   Min.   : 0.00   Min.   :1.00   Min.   :0.000
##  1st Qu.: 0.00   1st Qu.: 2.00   1st Qu.:1.00   1st Qu.:0.000
##  Median : 1.00   Median : 3.00   Median :2.00   Median :0.000
##  Mean   : 1.75   Mean   : 4.29   Mean   :1.87   Mean   :0.423
##  3rd Qu.: 3.00   3rd Qu.: 6.00   3rd Qu.:2.00   3rd Qu.:1.000
##  Max.   :13.00   Max.   :21.00   Max.   :5.00   Max.   :1.000
```

```
##  NA's   :26                    NA's   :5      NA's   :11
##   juge.enfant        place            abus         grav.cons
##  Min.   :0.000   Min.   :0.000   Min.   :0.000   Min.   :1.00
##  1st Qu.:0.000   1st Qu.:0.000   1st Qu.:0.000   1st Qu.:2.00
##  Median :0.000   Median :0.000   Median :0.000   Median :4.00
##  Mean   :0.277   Mean   :0.229   Mean   :0.278   Mean   :3.64
##  3rd Qu.:1.000   3rd Qu.:0.000   3rd Qu.:1.000   3rd Qu.:5.00
##  Max.   :1.000   Max.   :1.000   Max.   :1.000   Max.   :7.00
##  NA's   :5       NA's   :7       NA's   :7       NA's   :4
##     dep.cons        ago.cons       ptsd.cons        alc.cons
##  Min.   :0.000   Min.   :0.000   Min.   :0.000   Min.   :0.000
##  1st Qu.:0.000   1st Qu.:0.000   1st Qu.:0.000   1st Qu.:0.000
##  Median :0.000   Median :0.000   Median :0.000   Median :0.000
##  Mean   :0.397   Mean   :0.167   Mean   :0.216   Mean   :0.186
##  3rd Qu.:1.000   3rd Qu.:0.000   3rd Qu.:0.000   3rd Qu.:0.000
##  Max.   :1.000   Max.   :1.000   Max.   :1.000   Max.   :1.000
##
##    subst.cons       scz.cons           char            rs
##  Min.   :0.000   Min.   :0.0000   Min.   :1.00   Min.   :1.00
##  1st Qu.:0.000   1st Qu.:0.0000   1st Qu.:1.00   1st Qu.:1.00
##  Median :0.000   Median :0.0000   Median :1.00   Median :2.00
##  Mean   :0.265   Mean   :0.0826   Mean   :1.51   Mean   :2.06
##  3rd Qu.:1.000   3rd Qu.:0.0000   3rd Qu.:2.00   3rd Qu.:3.00
##  Max.   :1.000   Max.   :1.0000   Max.   :4.00   Max.   :3.00
##                                   NA's   :96     NA's   :103
##        ed             dr         suicide.s      suicide.hr
##  Min.   :1.00   Min.   :1.00   Min.   :0.00   Min.   :0.0
##  1st Qu.:1.00   1st Qu.:1.00   1st Qu.:0.00   1st Qu.:0.0
##  Median :2.00   Median :2.00   Median :0.00   Median :0.0
##  Mean   :1.87   Mean   :2.15   Mean   :0.79   Mean   :0.2
##  3rd Qu.:3.00   3rd Qu.:3.00   3rd Qu.:1.00   3rd Qu.:0.0
##  Max.   :3.00   Max.   :3.00   Max.   :5.00   Max.   :1.0
##  NA's   :107    NA's   :111    NA's   :41     NA's   :39
##   suicide.past     dur.interv
##  Min.   :0.000   Min.   :  0.0
##  1st Qu.:0.000   1st Qu.: 48.0
##  Median :0.000   Median : 60.0
##  Mean   :0.284   Mean   : 61.9
##  3rd Qu.:1.000   3rd Qu.: 75.0
##  Max.   :1.000   Max.   :120.0
##  NA's   :14      NA's   :50
```

Par contre, la variable `abus` est une variable binaire, qui prend les valeurs 0 et 1, et on pourrait dans certains cas vouloir la traiter comme un facteur en associant les modalités 0 et 1 aux étiquettes “Non” et “Oui” afin de faciliter la lecture des tableaux et des graphiques. Ceci peut se réaliser à l’aide de la commande `factor()`, comme dans l’exemple ci-dessous.

```r
smp$abus <- factor(smp$abus, levels = c(0,1), labels = c("Non", "Oui"))
```

Autre approche pour la gestion des data frame. Les packages data.table et dplyr offrent des commandes spécifiques pour lire des fichiers de données de type CSV et les représenter dans des structures identiques aux data frame standard de R, mais en apportant des petites améliorations pour leur manipulation.

2.3 Indexation d'éléments dans un data frame

On peut désigner n'importe quel objet dans un data frame par sa position en termes de n° de ligne et de n° de colonne. Par exemple, la profession (colonne n°2) du 3ème individu (ligne n°3) s'obtiendra ainsi :

```r
smp[3,2]
```

```
## [1] prof.intermediaire
## 8 Levels: agriculteur artisan autre cadre employe ... sans emploi
```

Il est également possible d'utiliser le nom des variables, à la place de leur numéro, par exemple `smp[3,"prof"]`, voire même le "rowname" de l'unité statistique d'intérêt, soit ici `smp["3","prof"]` puisque les "rowname" sont simplement constitués des numéros de rang des observations dans le data frame.

En d'autres termes, un data frame n'est rien d'autre qu'une structure tabulaire contenant des variables arrangées en colonnes qui portent des noms (`names()`, ou plus généralement `colnames()`) et des observations en lignes, elles-même nommées (`rownames()`). Considérons l'illustration ci-dessous (à gauche) :

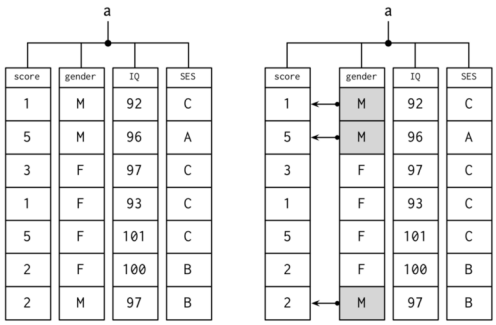

FIGURE 3 – Représentation schématique d'un data frame

La variable a est le nom d'un data frame contenant 4 variables : `score` (un score numérique sur une échelle allant de 1 à 5), `gender` (le sexe de l'individu, où `M` = homme et `F` = femme), `IQ` (le quotient intellectuel de l'individuel) et `SES` (le statut socio-économique de l'individu, codé en trois classes, `A`, `B` et `C`). le premier individu est donc un homme (`M`) de QI global 92 (`IQ`), dont la classe socio-économique (`SES`) est `C` et ayant un `score` de 1 point. Comme illustré dans la figure de droite, cette organisation par ligne/colonne permet d'associer à chaque individu l'ensemble des valeurs observées pour cet individu sur chaque variable. Qui plus est, il est possible de retourner les valeurs prises par une variable selon les valeurs observées sur une ou plusieurs autres variables. Par exemple, il est très simple de retourner une liste des scores de tous les individus dont le `sexe` vaut `M` en tapant simplement une instruction du type

```
a[a$gender == "M", "score"]
```

L'expression `a$gender == "M"` correspond à un test logique, et le résultat renvoyé, pour chaque élément testé, prend la valeur `TRUE` (vrai) ou `FALSE` (faux) selon que la condition est vérifiée ou non. Cela permet de retenir dans le data frame `a` que les lignes pour lesquelles la variable `gender` prend la valeur `M`. Ensuite, on filtre les colonnes de `a` en indiquant `"score"` comme nom de variable.

En fait, au lieu d'utiliser des numéros d'observation comme dans l'exemple précédent avec le data frame `smp`, on adresse ici des individus ou des lignes selon les valeurs observées pour certaines variables par ces mêmes individus. Ce principe général d'indexation est schématisé ci-dessous :

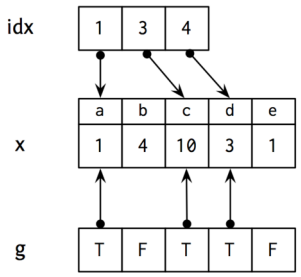

FIGURE 4 – Principe de la sélection indexée d'observations

Supposons donc que nous ayons une liste de trois numéros (1, 3, 4), stockés dans une variable appelée `idx`. Il peut s'agir, par exemple, de trois individus dont on connaît le numéro de position dans le tableau de données et pour lesquels on aimerait vérifier les données enregistrées. Les données qui nous intéressent sont celles de la variable `x`, qui contient 5 éléments (numérotés de 1 à 5, mais également nommées `a`, `b`, …, `e`, comme dans le cas des "rowname" d'un data frame). Alors, il est possible d'obtenir directement la 1ère, la 3ème et la 4ème valeur de `x` en utilisant l'une des notations suivantes : `x[c(1,3,4)]`, `x[idx]`, ou enfin `x[g]`. Dans cette dernière construction, on utilise une variable auxiliaire `g` ne contenant que des valeurs booléennes (`T` pour vrai et `F` pour faux) ; en d'autres termes, on ne demande à renvoyer les valeurs de `x` que lorsque `g` vaut `TRUE` (abbrégé `T`). Une formulation équivalente serait : `x[which(g == "T")]`. Voir l'aide en ligne de la commande `which()` pour décortiquer le résultat produit par cette commande.

3 Autres sources de données

3.1 Fichier binaire RData

Les fichiers R portent l'extension `.RData` ou `.rda` et peuvent être lus avec la commande `load()`. Ils sont généralement utiles pour sauvegarder des fichiers de données dans un format compressé (prenant moins de place sur le disque), et plus rapides à charger. On peut également utiliser ce format pour enregistrer n'importe quel objet R (une variable ou un tableau par exemple), voire plusieurs variables en même temps. Lorsque la commande `load()` a été exécutée, le nom de la ou des variables sauvegardées dans le fichier apparaissent dans l'espace de travail, ce que l'on peut vérifier en tapant :

```
ls()
```

C'est le format utilisé par R pour sauvegardé l'espace de travail lorsque l'on ferme une sessions R. Dans ce cas, R enregistre toutes les variables contenues dans l'espace de travail dans un fichier nommé `.RData` (c'est donc un

fichier masqué dans la plupart des explorateurs de fichiers). Lorsque l'on démarre R dans un certain répertoire de travail, si un tel fichier s'y trouve présent, il est automatiquement chargé par R.

En fait, plutôt que de laisser R enregistrer l'espace de travail à la fin de la session, on peut utiliser la commande `save.image()` pour sauvegarder l'espace de travail dans un fichier spécifique.

> **Historique des commandes.** Il est également possible de sauvegarder l'intégralité des commandes tapées durant une session à l'aide de la commande `savehistory()`.

3.2 Fichier SPSS et Stata

Le package `foreign` dispose de deux commandes permettant de charger des fichiers enregistrés au format SPSS (fichier `.sav`) ou Stata (fichier `.dta`) : `read.spss()` et `read.dta()`. Il est nécessaire de "charger le package" en tapant

```
library(foreign)
```

avant de pouvoir utiliser ces commandes. Dans le cas de SPSS, il est nécessaire de rajouter l'option `to.data.frame = TRUE` afin d'obtenir un data frame et non pas simplement une liste de variables.

3.3 Base de données relationnelles et autres formats

Certains packages permettent de se connecter directement sur des bases de données de type MySQL ou PostgreSQL (voire même MongoDB, Redis, ou sqlite). Dans ce cas, le mode d'interaction avec les données est légèrement différent car on utilise alors le langage de requête propre au langage, à moins d'utiliser des packages qui permettent d'assurer la conversion à partir des commandes R habituelles telles que `subset()`.

Il existe également des packages spécialisés dans le chargement et le traitement des fichiers de type XML, JSON, HDF5, etc. Généralement il suffit de chercher sur le site CRAN ou sur http://www.rseek.org.

4 Techniques plus avancées

Il existe bien d'autres méthodes pour interagir avec des tableaux de données sous R. Citons en particulier trois situations assez fréquentes :

- combiner ensemble différentes sources de données : pour associer deux tableaux de données disposant d'un identificateur commun (par exemple, une colonne avec des identifiants uniques pour les individus, le nom de variable dans chacun des tableaux n'étant pas nécessairement le même), on utilisera la commande `merge()` qui permet de fusionner deux data frame A et B en un seul et même data frame. Selon les options choisies, `all.x = TRUE`, `all.y = TRUE` ou `all = TRUE`, on conservera toutes les lignes du tableau A et les colonnes de B seront ajoutées à A même si certaines observations ne sont pas présentes dans B, ou c'est le tableau B qui servira de base de fusion, ou enfin toutes les observations de A et de B seront associées, même si elles ne sont pas en complètes correspondance entre les deux tableaux.
- aggréger des données : à partir de données individuelles, pour construire des données de synthèse (par exemple des moyennes pour chaque groupe) il est possible d'utiliser la commande `tapply()` ou `aggregate()`. Cette dernière présente l'avantage de retourner ses résultats sous forme de data frame, qu'il est possible d'exploiter ensuite pour continuer les analyses statistiques ou faire des représentations

graphiques. Le package plyr ou sa version plus récente, dplyr, fournit des options nettement plus améliorées pour ce type d'opérations.

— transformer des données : dans ce qui a été présenté sur les data frame, on considère explicitement que les valeurs prises par une variables sont regroupées dans une même colonne, et que donc chaque colonne représente des variables bien distinctes. Il arrive parfois que l'on stocke dans chaque colonne d'un tableau les valeurs observées pour chaque modalité ou niveau pris par une variable qualitative (l'exemple typique est une série de mesures répétées chez les mêmes individus à trois périodes différentes, et pour lequel on se retrouve avec un tableau à 4 colonnes, ou en plus d'une colonne d'identifiant unique pour les individus, on dispose de 3 colonnes regroupant les mesures collectées chez chaque individu pour une même période). Dans ce cas, le package `reshape2` permet de transformer ce type de tableau, dit en format "wide", en un data frame au format "long" comprenant trois colonnes : une colonne `id` désignant les identifiants individus, une colonne `variable` contenant les niveaux de la variable manipulée (par exemple, `période1`, `période2` et `période3`) et une colonne `value` contenant les mesures associées à chaque individu pour chacune des trois périodes.

Pour plus d'informations, il peut être utile de consulter l'une des références suivantes :

1. Spector, P (2008). *Data Manipulation with R*. Springer
2. R Cookbook / Manipulating Data
3. Muenchen, B. R for SAS and SPSS Users