# Introduction au logiciel R

# Manipuler des données

Laurence Viry

MaiMoSiNE - Collège des écoles doctorales Grenoble

7-16 Février 2017

# Plan du cours

# Introduction

En statistique, les données constituent **le point de départ de toute analyse**, un premier travail de mise en forme des données est presque toujours indispensable. Il faudra savoir maitriser des opérations comme:

- l'**importantion de données** sous différents formats,
- **exporter des données et des résultats** sous différents formats,
- **concaténer** ou **extraire** des données,
- repérer les individus ayant des **données manquantes** ou **aberrantes**.
- **changer le type** de certaines varaibles,
- ...

**R** fournit des **outils** et des **capacités de programmation** pour effectuer ces différentes tâches.

# Importer des données

- R lit des données dans des fichiers **ASCII** avec les fonctions **read.table()**, **read.csv()**,**scan()**,**read.fwf()**,...
- R peut également lire des fichiers dans d'**autres formats** (Excel, SAS, SPSS, . . .) et accéder à **des bases de données**.
- R utilise le **répertoire de travail** (getwd()), sinon il est nécessaire de préciser le chemin d'accès ou changer de répertoire de travail (setwd()).

# Importer des données

Les **données** sont initialement **collectées** et éventuellement **traitées** par un **logiciel**, un **tableur** ou un **logiciel de statistiques**, **chaque logiciel ayant son propre format**.

On échangera les données soit dans un **format commun à tous**, un format texte (**.csv** par exemple) soit en utilisant un package qui permet de lire les **formats propriétaires des autres logiciels**, le choix dépendant du contexte et du volume des données.

**Les avantages des fichiers csv**:

- Peut être lu par n'importe quel logiciel passé, présent et probablement futur,
- Pour la compatibilité entre plate-forme (Windows, Mac, Linux),
- Pour la facilité de lecture par un être humain comparativement à d'autres formats tels que XML, HL7, JSON etc.

Pas forcément adapté aux gros volumes de données pour son **volume de stockage** et **la rapidité de lecture**.

# Lecture fichier texte : read.table

Les données sont contenus dans un **fichier** avec des **individus en ligne** et des **variables en colonnes**, elles sont séparées par un séparateur de colonnes.

- Les donnés se trouvent dans le fichier **don.csv** qui est dans un répertoire "**data**" directement en dessous du répertoire de travail

```
> table <- read.table("data/don.cvs",sep=";",header=TRUE,
+                     dec=",",row.names=1)
```

- **Le chemin peut-être une URL**:
  decath <- read.table("http://www.agrocampus-ouest.fr/math/livreR/decathlon.csv",header=TRUE,row.names=1)

- **un caractère spécial** peut indiquer qu'il y a **des données manquantes**:

```
> table2 <- read.table("data/don2.csv",sep=";",header=TRUE,
+                      na.strings="*")
```

- Le **résultat** de l'importation est de type **data-frame**.

Effectuons un premier traitement pour valider la lecture:

```
> summary(table)
```

# Fonctions utiles dans un data-frame

- **head**() - pour voir les 6 premières lignes
- **tail**() - pour voir les 6 dernières lignes
- **dim**() - ses dimensions
- **nrow**() - le nombre de lignes
- **ncol**() - le nombre de colonnes
- **str**() - structure de chaque colonne
- **names**() - liste l'attribut names d'un data frame (ou n'importe quel autre objet), ce qui donne les noms des colonnes.

Les données sont dans **<répertoire de travail>/data**

**Sans données manquantes**:

```
individu;taille,poids;pointure;sexe
Alice;184;80;44;M
Alexis;175;78;43;M
Marcel ; 158;72;42;M
```

```
> table <- read.table("data/don.cvs",sep=";",header=TRUE,
+                     dec=",",row.names=1)
> summary(table)
```

**Le caractère "*" représente les données manquantes**:

```
individu;taille,poids;pointure;sexe
Alice 184 80 44 M
Alexis 175 * 43 M
Marcel  158 72 * M
```

```
> table2 <- read.table("data/don2.csv",sep=" ",header=TRUE,
+     ,na.strings="*")
> summary(table2)
```

# La fonction scan

La fonction **scan** est plus flexible que **read.table**.

- Une différence est qu'il est **possible de spécifier le mode des variables**:

  ```
  > mydata <- scan("data.dat", what = list("", 0, 0))
  ```

  Dans cet exemple, **scan** lit trois variables, la première de mode caractère et les deux suivantes sont de mode numérique.

  "**myData**" est **une liste** de 3 vecteurs.

- **scan()** peut être utilisée pour **créer des objets de mode différent** (vecteurs, matrices, tableaux de données, listes,...).

  Par défaut, c'est-à-dire **si what est omis**, **scan() crée un vecteur numérique**.

Pour en savoir plus **help(scan)**

# Formats propriétaires et Base de données

R peut également lire des fichiers dans d'**autres formats** (Excel, SAS, SPSS, . . .) et accéder à **des bases de données**.

- Le package **foreign** permet d'**importer des données** en format propriétaire **binaire** tels que **Stata**, **SAS**, **SPSS**, etc.

```
library(foreign)
read.dta("calf_pneu.dta") # for Stata files
read.xport("file.xpt") # for SAS XPORT format
read.spss("file.sav") # for SPSS format
read.mpt("file.mtp") # for Minitab Portable Worksheet
```

Une autre solution pour des **fichiers SPSS**

```
library(Hmisc)
spss.get()
...
```

- Il y a plusieurs packages permettant de connecter R à un **DBMS** (**RODBC**, **RMySQL**, **RSQLite**, **ROracle** etc.).

# Sauvegarder des données ou des résultats

Une fois les analyses effectuées et les résultats obtenus, il est souvent important de **les sauvegarder pour les communiquer** à d'autres personnes ou d'autres logiciels ou **les réutiliser dans d'autres analyses**.

- **Le format texte** est fréquemment le format utilisé en utilisant la fonction **write.table**().

  **write.table**(tablo_res,"monfichier.csv",sep=";",row.names=FALSE)

- **Exporter les résultats et/ou les données dans un fichier binaire** suffixé **.Rdata** que R sera capable de décrypter par la suite grâce à la fonction **save**():

  **save**(file="nom_fichier",<liste des variables>)

```
> x <- runif(20)
> y <- list(a = 1, b = TRUE, c = "oops")
> save(x, y, file = "xy.RData")
```

- L'**utilisation des fichiers sauvegardées** par la commande **save**() se fait par la commande **load**().

```
> load(file="xy.RData")
```

# Manipuler des données

- **Changer le type d'une variable**: à l'issue d'une importation, une variable qualitative dont les modalitées sont numériques sera comprise par R comme une variable quantitative,...
- **Découper en classes une variable qualitative**: le passage d'une variable quantitative à une variable qualitative est fréquemment nécessaire en statistiques pour l'adapter à la méthode utilisée (AFC,AFCM,...)
- **Modifier les niveaux d'une variable qualitative**: fusionner un ou plusieurs niveaux en fonction des effectifs,...
- **Repérer les données manquantes**: permettre la prise en compte des données manquantes dans le traitement statistique des données.
- **Repérer les données aberrantes**: permettre la prise en compte des données aberrantes dans le traitement statistique des données.
- ...

# Changer de type une variable

Il est souvent nécessaire de changer la classe d'une variable. A l'importation, une variable qualitative est comprise par R comme une variable quantitative,...

- **Qualitatives ou quantitatives?** 3 méthodes:

```
> X <-c(rep(5,2),rep(12,4),13)
> X
[1]  5  5 12 12 12 12 13
> is.factor(X)
[1] FALSE
> is.numeric(X)
[1] TRUE
> summary(X)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   5.00    8.50   12.00   10.14   12.00   13.00
```

- Le **passage de quantitatives à qualitative** est simple:

```
> Xqual <- factor(X)
> Xqual
[1] 5  5  12 12 12 12 13
Levels: 5 12 13
```

# Variable quantitative vers variable qualitative

- Le **passage de qualitative à quantitatives avec recodage des modalités**:
  1 seule étape:

```
> Xqual
[1] 5  5  12 12 12 12 13
Levels: 5 12 13
> as.numeric(Xqual)
[1] 1 1 2 2 2 2 3
```

- Le **passage de qualitative à quantitatives sans recodage des modalités**:
  2 étapes

```
> Xqual
[1] 5  5  12 12 12 12 13
Levels: 5 12 13
> prov <- as.character(Xqual)
> prov
[1] "5"  "5"  "12" "12" "12" "12" "13"
> as.numeric(prov) # as.numeric(as.character(Xqual))
[1]  5  5 12 12 12 12 13
```

# Découpage en classes

Le découpage en classes d'une variable quantitatives peut se faire avec **deux approches**:

- **Les seuils des classes sont choisis par l'utilisateur**: pour définir ces seuils de façon automatique, on utilisera la fonction **cut**. Les classes sont de la forme $]a_i, a_{i+1}]$:

```
> set.seed(654)  # on fixe la graine du générateur
> X <- rnorm(n=100,mean=0,sd=1)
> # Découpage en 3 noveaux: [min(X),-0.2],[-0.2,0.2],[0.2,max(X)
> Xqual <- cut(X,breaks = c(min(X)-1e-10,-0.2,0.2,max(X)))
> class(Xqual)
[1] "factor"
> summary(Xqual)  # ou table(Xqual)
(-3.27,-0.2]   (-0.2,0.2]   (0.2,1.69]
          41           20           39
```

- **Un découpage automatique proposant des effectifs équivalents dans chaque classes**

# Découpage en classes (suite)

- **Un découpage automatique proposant des effectifs équivalents dans chaque classes**: si nous voulons des effectifs équilibrés dans chacune des trois modalités, on utilisera la fonction **quantile**.

```
> decoupe <- quantile(X,probs=seq(0,1,length=4))
> decoupe[1] <- decoupe[1]-1e-10
> Xqual <- cut(X,decoupe)
> table(Xqual)

Xqual
(-3.27,-0.437] (-0.437,0.399]   (0.399,1.69]
            34              33             33
```

# Modifier le niveau des facteurs

## Fusionner un ou plusieurs niveaux

```
> table(Xqual)

Xqual
(-3.27,-0.437] (-0.437,0.399]   (0.399,1.69]
            34              33             33

> levels(Xqual) <- c(1,2,3) # modifier les labels des modalités
> table(Xqual)

Xqual
 1  2  3
34 33 33

> # fusionner la modalité 1 et 3
> levels(Xqual) <- c(1,2,1)
> table(Xqual)

Xqual
 1  2
67 33
```

# Modifier le niveau des facteurs

- **Niveau de référence**: pour certaines méthodes, il faudra tenir compte de l'ordre d'apparition des niveaux ou spécifier une niveau de référence (analyse de variance,...), nous utiliserons la fonction **relevel**:

```
> X <- c(1,2,1,3,2,2,1)
> Xqual <- factor(X,label=c("classique","nouveau","placebo"))
> Xqual2 <- relevel(Xqual,ref="placebo")
```

- **Contrôler l'ordre des niveaux**: recréer un facteur à partir du facteur existant en spécifiant l'ordre des niveaux.

```
> Xqual3 <- factor(Xqual,levels=c("placebo", "nouveau","classiqu
> Xqual3 <- Xqual3[-4] # élimine l'individu avec la modalité "3"
> table(Xqual3) # la modalité "3" n'apparait plus
Xqual3
  placebo   nouveau classique
        0         3         3
> # élimine la modalité "3"
> Xqual3 <- factor(as.character(Xqual3)) # élimine la modalité "
>
```

# Repérer les individus manquants

Dans R, les **données manquantes** sont représentées par **NA**. La fonction **is.na** permet de les retrouver.

```
> X <- rnorm(10,0,1)
> X[c(2,7,10)] <- NA
> summary(X)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
-1.3630 -0.5980 -0.3101 -0.1073  0.2859  1.5460       3
> mean(X)
[1] NA
> mean(X,na.rm=TRUE)
[1] -0.1073282
> selectNA <- is.na(X)
> selectNA
 [1] FALSE  TRUE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE  TRUE
> which(selectNA) # Quels sont les indices correspondants
[1]  2  7 10
> X2 <-X[!selectNA] # On élimine les individus correspondants
```

# Repérer les individus manquants dans un tableau de données

```
> Y <- factor(c(rep("A",3),NA,rep("M",4),"D",NA))
> don <-data.frame(X,Y)
> summary(don)
       X                Y
 Min.   :-1.3629   A   :3
 1st Qu.:-0.5980   D   :1
 Median :-0.3101   M   :4
 Mean   :-0.1073   NA's:2
 3rd Qu.: 0.2859
 Max.   : 1.5458
 NA's   :3
> selectNA  <- is.na(don)
> #
> # au moins une donnée manquante
> aelim_any <- apply(selectNA, MARGIN=1,FUN=any)
> aelim_any
 [1] FALSE  TRUE FALSE  TRUE FALSE FALSE  TRUE FALSE FALSE  TRUE
```

# Repérer les individus manquants sur un tableau de données

```
> # toutes les données manquantes
> aelim_all <- apply(selectNA, MARGIN=1,FUN=all)
> aelim_all
 [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE
> don2 <-don[!aelim_all,] # individus éliminés
> which(is.na(don))
[1]  2  7 10 14 20
> which(is.na(don),arr.ind=T) # option arr.ind (array indices) de wh
     row col
[1,]   2   1
[2,]   7   1
[3,]  10   1
[4,]   4   2
[5,]  10   2
```

# Repérer les individus aberrants

On utilise la fonction **boxplot** qui fournit dans sa composante **out** les valeurs aberrantes.

```
> library(rpart)
> data("kyphosis")
> names(kyphosis)
```

```
[1] "Kyphosis" "Age"      "Number"   "Start"
```

```
> boxNumber <- boxplot(kyphosis[,"Number"]) # repère
> #                                    les individus aberrants
> valaberrante <- boxNumber$out # valeurs aberrantes
> which(kyphosis[,"Number"]%in%valaberrante) # quels individus ?
```

```
[1] 43 53
```

# Concaténer des tableaux de données

>

# Tableaux de contingences

```
> tension <- factor(c(rep("Faible",5),rep("Forte",5)))
> tension

 [1] Faible Faible Faible Faible Faible Forte  Forte  Forte  Forte
Levels: Faible Forte

> laine <- factor(c(rep("Mer",3),rep("Ang",3),rep("Tex",4)))
> laine

 [1] Mer Mer Mer Ang Ang Ang Tex Tex Tex Tex
Levels: Ang Mer Tex

> # fusionnons ces deux variables dans un data.frame
>  don <-data.frame(tension, laine)  # cbind.data.frame(tension, lai
>  # Tableau de contingences
>  tabcroise <-table(don$tension,don$laine)
>  tabcroise

        Ang Mer Tex
  Faible   2   3   0
  Forte    1   0   4
```

# Tableaux croisés

Lorsqu'on a **deux variables qualitatives** observées sur un échantillon, les données peuvent être présentées sous deux formes:

- **Tableau de contingence**:

```
> tension <- factor(c(rep("Faible",5),rep("Forte",5)))
> tension
 [1] Faible Faible Faible Faible Faible Forte  Forte  Forte  For
Levels: Faible Forte
> laine <- factor(c(rep("Mer",3),rep("Ang",3),rep("Tex",4)))
> laine
 [1] Mer Mer Mer Ang Ang Ang Tex Tex Tex Tex
Levels: Ang Mer Tex
> # fusionnons ces deux variables dans un data.frame
>  don <-data.frame(tension, laine)  # cbind.data.frame(tension,
>  # Tableau de contingences
>  tabcroise <-table(don$tension,don$laine)
```

- **Tableaux individus X variables**

```
> tabframe <- as.data.frame(tabcroise)
```

# Tableaux croisés

- **Tableau de contingence**:
- **Tableaux individus X variables**

```
>  tabframe <- as.data.frame(tabcroise)
>  tabframe
```

```
    Var1 Var2 Freq
1 Faible  Ang    2
2  Forte  Ang    1
3 Faible  Mer    3
4  Forte  Mer    0
5 Faible  Tex    0
6  Forte  Tex    4
```

Nous obtenons **une fréquence** pour **chaque combinaison** et non pas une ligne par individu.