Chapitre 4 Mon premier jeu de données

4.1 Les types de valeurs dans R

Une valeur constitue l'unité de base des données pour R. Comme pour la plupart des logiciels, elles peuvent être de trois **types** :

- **Numérique** : entier, double
- **Caractère** : texte ou code
- **Logique** : booléens

Ces valeurs sont ensuites structurées au sein de vecteur qui sont compilés dans des *dataframes*.

4.2 Avant de manipuler le data.frame : installer et charger un package

Pour manipuler notre base de données, nous allons nous servir de fonctionnalités présentes dans le package `tidyverse` .

La première fois que l'on veut utiliser un package, il faut le télécharger, avec `install.packages()` :

```
install.packages ("tidyverse")
```

Puis à l'ouverture de R, le charger (si on a besoin de ses fonctions), avec `library()` :

```
library ("tidyverse")
```

Le package est prêt à être utilisé !

4.3 Importation de données

Conseil : exporter les données au format .csv ; c'est le format le plus interopérable (supporté par tous les logiciels stat). Utiliser ensuite la fonction `read.csv` après avoir défini le répertoire de travail (ou en donnant le chemin complet)

```
base <- read.csv ("Data/Base_synth_territoires.csv",
                  header = T, sep=";", dec=",")
```

- `header=` indique la présence des noms de variable sur la première ligne
- `sep=` indique le séparateur de champ : ';' '/t' pour tabulation.
- `dec=` indique le séparateur de décimale ('.' par défaut)
- `colClasses=` permet de préciser le type de la données en entrée

⇒ `?read.csv` pour plus d'options

Autres façons d'importer les données

- Fonction `read_delim` , du package `readr` , plus rapide
- Fonction `fread` , du package `data.table` , beaucoup plus rapide !!
- Pour importer les fichiers XLS, ODT ou DBF, il existe des fonctions et des packages spécifiques
- **Le passage par un fichier csv est très recommandé**

4.4 Structure des données : le *dataframe*

	V1	V2	V3	V4	V5	V6	...	Vp
1								
2								
3								
4								
5								
6								
7								
8								
9								
10								
11								
12								
⋮								
n								

- n lignes (observations)
- p colonnes (variables)

```
nrow (base)
```

```
## [1] 36689
```

```
ncol (base)
```

```
## [1] 38
```

```
dim (base)
```

```
## [1] 36689    38
```

4.5 Gérer le type des variables

À chaque type de variable (numéric, factor,…) correspond une utilisation. Lors de l'import de données, un type est affecté automatiquement par R. Mais le type peut être erronné. Il suffit alors de de les convertir :

```
base <- mutate  (base, LIBGEO = as.character (LIBGEO))
```

ou `as.factor` , `as.numeric()` , etc...

La fonction `typeof()` permet de connaître le type de la variable.

Pour être sûr de ne pas faire de bêtise, il vaut mieux gérer les types au moment de l'importation avec le paramètre `colClasses` → exercice !

4.6 Les valeurs particulières

- NA : Valeur manquante (*Not Available*)
- NaN : Pas un nombre (*Not a Number*, e.g., 0/0)
- -Inf, Inf : Infini positif ou négatif, e.g. 1/0

```
V1 <- c (1 , 14 , NA , 32.7)
mean (V1)              # renvoie NA. Not good !
```

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

```
mean (V1, na.rm = T)   # renvoie 15.9   - OK !
```

```
## [1] 15.9
```

⇒ Le module 2 "préparation des données" aborde la gestion de ces valeurs particulières

4.7 Exercice : Importer les données et premier coup d'oeil

On peut importer n'importe quel format de données en R (Excel, SAT, Stata, SQL…) → module 2. Pour ce module, nous ne voyons que l'importation de fichier .CSV. Si vous avez une base de données en Excel ou LibreOffice Calc, sauvegardez l'onglet que vous souhaitez en faisant "enregistrer sous" → "délimité CSV". Ici, nous travaillerons sur une base de données communales fournie par l'Insee, dite "comparateur de territoires". Le fichier source (Excel) figure dans le sous-répertoire "Data" et contient toutes les métadonnées.

- Utiliser la fonction read.csv pour importer ce fichier et stockez le dans un objet df. **Veillez à ce que la région soit bien importée comme un facteur et non un entier**.
- Inspecter le dataframe avec les fonctions vues auparavant … et plus !

```
df <- read.csv (file = "Data/Base_synth_territoires.csv",
                sep = ";", dec = ",",
                colClasses = c (NA, "NULL", "character", "NULL","NULl
                                NA, NA, NA, NA, rep ("NULL", 28)))

str (df)
```

```
## 'data.frame':    36689 obs. of  6 variables:
##  $ CODGEO  : Factor w/ 36689 levels "01001","01002",..: 1 2 3 4 5
##  $ REG     : chr  "84" "84" "84" "84" ...
##  $ P14_POP : int  767 239 14022 1627 109 2570 743 338 1142 397 ...
##  $ P09_POP : int  787 207 13350 1592 120 2328 660 336 960 352 ...
##  $ SUPERF  : num  15.95 9.15 24.6 15.92 5.88 ...
##  $ NAIS0914: int  40 16 1051 117 8 175 59 12 56 25 ...
```

On peut également gérer les problèmes d'encodage de caractères (si on passe de Windows à un système Unix, par exemple)

```
head (df)
```

```
##   CODGEO REG P14_POP P09_POP SUPERF NAIS0914
## 1  01001  84     767     787  15.95       40
## 2  01002  84     239     207   9.15       16
## 3  01004  84   14022   13350  24.60     1051
## 4  01005  84    1627    1592  15.92      117
## 5  01006  84     109     120   5.88        8
## 6  01007  84    2570    2328  33.55      175
```

```
tail (df)
```

```
##       CODGEO REG P14_POP P09_POP SUPERF NAIS0914
## 36684  97419  04    6722    6822 177.60      519
## 36685  97420  04   22406   22437  57.84     1963
## 36686  97421  04    7132    7406 103.82      661
## 36687  97422  04   76796   72658 165.43     6663
## 36688  97423  04    7198    7057  42.58      531
## 36689  97424  04    5295    5989  84.40      430
```

```
names (df)
```

```
## [1] "CODGEO"   "REG"      "P14_POP"  "P09_POP"  "SUPERF"   "NAIS0
```

```
class (df) # la classe de l'objet (du point de vue langage orienté ol
```

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

```
typeof (df) # le type d'objet du point de vue "interne" à R
```

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

```
# Accéder a une variable directement (on obtient un vecteur) :
df$CODGEO [1:10] # Captain crochet poweeeeerr !!!
```

```
##  [1] 01001 01002 01004 01005 01006 01007 01008 01009 01010 01011
## 36689 Levels: 01001 01002 01004 01005 01006 01007 01008 01009 010
```

```
class (df$CODGEO)
```

```
## [1] "factor"
```

```
typeof (df$CODGEO)
```

```
## [1] "integer"
```

```
length (df)
```

```
## [1] 6
```

```
length (df$REG)
```

```
## [1] 36689
```