Guide R

Prof. Audrey Bürki, Samuel Arthers, Mégane Bollenrücher

2024-09-19

Contents

Merci de prendre note que ce bookdown est en cours de rédaction.

Chapter 1

Introduction

Ceci est le guide R que nous proposons pour vous accompagner durant le cours de Statistiques I.

Chapter 2

Installation et environnement R et Rstudio

2.1 Installation

2.2 Environnement de travail

Chapter 3

Objects et opérateurs

3.1 Objects dans R

Une variable permet de stocker une valeur ou un objet dans R. De cette façon, il sera possible d'accéder à la valeur ou à l'objet qui est stocké dans la variable.

```
# Assignation de la valeur 3 à la variable "ma_variable"
ma_variable <- 3
```

La ligne de code ci-dessus déclare une variable nommée "ma_variable" et lui assigne la valeur de 3. L'exécution de ce code n'affiche pas de résultat dans la console, mais l'objet nommé "ma_variable" est bien créée et stockée dans l'environnement.

Pour afficher le contenu de la variable, il suffit de taper le nom de celle-ci pour l'afficher dans la console.

```
# Affichage du contenu de la variable "ma_variable"
ma_variable
```

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

De plus, il est également possible d'utiliser la fonction `print()` qui permet d'afficher la valeur ou l'objet de la variable sélectionnée.

```
# Affichage du contenur de la variable "ma_variable"
print(ma_variable)
```

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

En langage R, il existe différents types d'objets qui peuvent être assignés à des variables comme les scalaires, les vecteurs, les facteurs, les matrices et les bases de données. Ces objets sont présentés dans les points suivants.

3.1.1 Scalaire

Un scalaire permet de stocker un objet sous forme de valeur numérique, de chaîne de caractères ou de valeur logique.

```
# Scalaire numerique
a <- 3
a
```

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

Une chaîne de caractères est une suite de caractères qui doit être écrit entre guillemets (" ").

```
# Scalaire sous forme d'une chaine de caractères
b <- "Statistique"
b
```

```
## [1] "Statistique"
```

Une valeur logique est une quantitée binaire (vrai ou faux). Ces variables s'écrivent `TRUE` et `FALSE`. Il est également possible d'utiliser `T` et `F` comme abréviations.

```
# Scalaire logique
c <- TRUE
c
```

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

```
d <- F
d
```

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

3.1.2 Vecteur

Un vecteur est un objet qui permet de stocker une liste ordonnée d'éléments. Les éléments d'un vecteur doivent être du même type. Pour pouvoir stocker une information dans un seul objet, il faut utiliser la fonction `c()` qui permet de combiner les arguments de la fonction.

```
# Vecteur numérique
a <- c(1, 2, 3, 4, 5, 6)
a
```

```
## [1] 1 2 3 4 5 6
```

```
# Vecteur sous forme d'une chaine de caractères
b <- c("Un", "Deux", "Trois", "Quatre", "Cinq", "Six")
b
```

```
## [1] "Un"     "Deux"   "Trois"  "Quatre" "Cinq"   "Six"
```

```r
# Vecteur logique
c <- c(TRUE, FALSE, TRUE, FALSE, FALSE, TRUE)
c
```

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

Étant donné que le vecteur est un objet ordonné, il est possible d'accéder, remplacer ou modifier un ou plusieurs éléments par rapport à leur position dans l'objet. Il est nécessaire d'utiliser les crochets `[ ]` pour indiquer le ou les éléments à manipuler.

Pour accéder un seul élément du vecteur, il suffit d'écrire le nom de la variable suivi de crochets contenant la position de l'élément sélectionné.

```r
# Extraction d'un élément:
b[2]
```

```
## [1] "Deux"
```

Il est possible d'accéder à plusieurs éléments en mettant un vecteur de position entre crochets.

```r
# Extraction de plusieurs éléments:
b[c(2,3,5)]
```

```
## [1] "Deux"  "Trois" "Cinq"
```

Il est aussi possible d'accéder à une série d'éléments à la suite en mettant le signe `:` entre les 2 positions désirées.

```r
# Extraction d'une série d'éléments:
b[2:5]
```

```
## [1] "Deux"   "Trois"  "Quatre" "Cinq"
```

Il est également possible d'accéder à certaines lignes en fonction de la valeur logique d'un vecteur ayant la même taille du vecteur sélectionné. Si la position est mise à `TRUE`, la valeur sera sélectionnée et dans le cas dans lequel la valeur est mise à `FALSE` la valeur ne sera pas retenue.

```r
# Extraction de plusieurs éléments en fontion de la valeur logique:
b[c(FALSE, TRUE, TRUE, FALSE, TRUE, TRUE)]
```

```
## [1] "Deux"  "Trois" "Cinq"  "Six"
```

3.1.3 Facteur

Un facteur est un vecteur dont les éléments peuvent prendre que des valeurs prédéfinies. Un facteur dispose de l'argument `levels` qui permet de définir des catégories de valeurs. Le facteur est généralement utilisé pour stocker des variables catégorielles.

Pour commencer, il faut définir un vecteur qui peut être numérique, logique ou chaine de caractères. Le facteur est une variable nominale.

```
genre <- c("Homme", "Femme", "Femme", "Femme", "Homme")
genre
```

```
## [1] "Homme" "Femme" "Femme" "Femme" "Homme"
```

La fonction `factor()` permet de créer un facteur à partir d'un vecteur.

```
genre <- factor(genre)
genre
```

```
## [1] Homme Femme Femme Femme Homme
## Levels: Femme Homme
```

On note que la sortie est légèrement différente lorsque le vecteur est mis sous forme de facteur à 2 niveaux (Femme, Homme). Ceci s'affiche à la ligne `Levels`. Par défaut, les niveaux d'un facteur sont affichés par ordre alphabétique et numérique croissant. Il est possible de fixer l'ordre en ajoutant l'argument `levels` en appliquant la fonction `factor()`.

```
sexe <- c("H", "F", "F", "F", "H")
sexe <- factor(sexe, levels = c("H", "F"))
sexe
```

```
## [1] H F F F H
## Levels: H F
```

Dans le cas dans lequel on aimerait modifier un élément, il n'est pas possible d'affecter une valeur qui n'est pas défini comme un niveau. On voit donc apparaître une erreur dans la console.

```
genre[2] <- "Fille"
```

```
## Warning in `[<-.factor`(`*tmp*`, 2, value = "Fille"): niveau de facteur
## incorrect, NAs générés
```

Il est possible de renommer les niveaux en utilisant la fonction `levels()`. **Il faut faire attention à l'ordre lorsqu'on utilise la fonction `levels()`.** L'argument `order` permet d'ordonner les labels proposés. Le facteur est dès lors une variable ordinale.

```
levels(genre) <- c("Fille", "Garcon")
genre
```

```
## [1] Garcon <NA>   Fille  Fille  Garcon
## Levels: Fille Garcon
```

Il est possible d'avoir un facteur numérique en y affectant une catégorie avec l'argument `labels` lors de l'utilisation de la fonction `factor()`.

```
satisfaction <- factor(c(3, 3, 4, 1, 2, 1, 1), labels = c("Pas du tout d'accord", "Pas d'accord",
satisfaction
```

```
## [1] D'accord             D'accord             Tout à fait d'accord
## [4] Pas du tout d'accord Pas d'accord         Pas du tout d'accord
## [7] Pas du tout d'accord
## 4 Levels: Pas du tout d'accord < Pas d'accord < ... < Tout à fait d'accord
```

3.1.4 Matrice

Une matrice est un vecteur dont les éléments sont disposés sous forme d'un tableau qui comporte des lignes et des colonnes. De façon équivalente au vecteur, les éléments de la matrices doivent être de même classe (numérique, logique ou chaîne de caractères). La fonction `matrix()` permet de déclarer une matrice. Il faut ajouter l'argument `ncol` et/ou `nrow` pour déterminer la forme de la matrice.

```
A <- matrix(1:24, nrow=6, ncol=4, byrow=FALSE)
A
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    1    7   13   19
## [2,]    2    8   14   20
## [3,]    3    9   15   21
## [4,]    4   10   16   22
## [5,]    5   11   17   23
## [6,]    6   12   18   24
```

Par défaut, le remplissage se fait par colonne. Il faut donc mettre l'argument `byrow` à `TRUE` pour remplir la matrice par ligne.

```
B <- matrix(1:24, nrow=6, ncol=4, byrow=TRUE)
B
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    1    2    3    4
## [2,]    5    6    7    8
## [3,]    9   10   11   12
## [4,]   13   14   15   16
## [5,]   17   18   19   20
## [6,]   21   22   23   24
```

L'objet matrice dispose de la fonction `dim()` qui permet d'obtenir sa dimension. Le premier terme correspond aux nombres de lignes et le deuxième correspond aux nombres de colonnes.

```
dim(B)
```

```
## [1] 6 4
```

Les fonctions `rownames()` et `colnames()` permettent de récupérer ou de définir les noms des lignes et des colonnes. Attention de bien mettre le bon nombre de noms aux lignes et aux colonnes.

```
rownames(B) <- c("L1", "L2", "L3", "L4", "L5", "L6")
colnames(B) <- c("C1", "C2", "C3", "C4")
B
```

```
##    C1 C2 C3 C4
## L1  1  2  3  4
## L2  5  6  7  8
## L3  9 10 11 12
## L4 13 14 15 16
## L5 17 18 19 20
## L6 21 22 23 24
```

Comme pour le vecteur, il est possible d'accéder à un ou plusieurs éléments de la matrice. Pour extraire une ligne de la matrice, il faut utiliser les crochets avec une virgule pour délimiter les deux dimensions de la matrice`[ , ]`. Le premier terme (celui avant la virgule) permet d'accéder aux colonnes et le deuxième terme (celui après la virgule) permet d'accéder aux lignes. Pour accéder à un seul élément, il faut indiquer la position de la ligne et de la colonne désirée.

```
# Extraction d'un seul élément
A[2,3]
```

```
## [1] 14
```

Pour accéder à une ligne complète, il suffit de mettre la position de la ligne désirée avant la virgule.

```
# Extraction d'une ligne
A[2, ]
```

```
## [1]  2  8 14 20
```

Pour accéder à une colonne complète, il suffit de mettre la position de la colonne désirée après la virgule.

```
# Extraction d'une colonne
A[, 3]
```

```
## [1] 13 14 15 16 17 18
```

Pour accéder à un groupe d'élément, il faut déterminer l'intervalle des lignes et des colonnes désirées.

```
# Extraction de quelques éléments regroupées
A[3:5, 2:3]
```

```
##      [,1] [,2]
## [1,]    9   15
```

```
## [2,]   10   16
## [3,]   11   17
```

3.1.5 Dataframe

Un jeu de données se structure sous forme d'un tableau dans lequel chaque ligne correspond à une observation (individu) et chaque colonne à une caractéristique (variable). Les data frame sont les objets les plus utilisées lors de l'analyse d'une base de données. Contrairement aux vecteurs et aux matrices, une dataframe peut avoir différents type de variables (numérique, logique et chaînes de caractères). La fonction `data.frame()` permet la création de la base de données.

```
dataframe <- data.frame(
  ID = 1:5,
  Genre = c("Homme", "Femme", "Femme", "Femme", "Homme"),
  Age = c(45, 42, 45, 43, 44)
)
dataframe
```

```
##   ID Genre Age
## 1  1 Homme  45
## 2  2 Femme  42
## 3  3 Femme  45
## 4  4 Femme  43
## 5  5 Homme  44
```

Les colonnes d'une dataframe sont toujours nommées et correspondent à la variable mesurée. Les lignes sont automatiquement numérotées par ordre.

La fonction `str()` permet d'afficher la structure de la dataframe en affichat le nom de la variable, le type de celle-ci ainsi que les valeurs des observations.

```
# Structure
str(dataframe)
```

```
## 'data.frame':    5 obs. of  3 variables:
##  $ ID   : int  1 2 3 4 5
##  $ Genre: chr  "Homme" "Femme" "Femme" "Femme" ...
##  $ Age  : num  45 42 45 43 44
```

La fonction `View()` permet de visionner la data frame dans une autre fenêtre.

```
# Structure
View(dataframe)
```

Afin d'analyser les données, il est important de pouvoir d'en extraire uniquement une partie. Il existe deux façons d'extraire une colonne. La première consiste à reproduire le cas de la matrice en séléctionnant la position de la colonne.

```
# Extraction de colonnes
dataframe[, 2]
```

```
## [1] "Homme" "Femme" "Femme" "Femme" "Homme"
```

La deuxième option est d'utiliser le symbole `$`. Il doit être placé entre le nom de la data frame et le nom de la colonne. **Il est conseillé d'utiliser cette option pour extraire une colonne d'une data frame.**

```
# Extraction de colonnes
dataframe$Genre
```

```
## [1] "Homme" "Femme" "Femme" "Femme" "Homme"
```

Pour extraire une ligne de la base de donnée, il faut procéder comme pour la matrice.

```
# Extraction de ligne
dataframe[2, ]
```

```
##   ID Genre Age
## 2  2 Femme  42
```

Pour extraire les observations (lignes) qui possèdent certaines caractéristiques, il est possible d'écrire la ligne suivante comme suit:

```
# Extraction de ligne
dataframe[dataframe$Genre == "Homme", ]
```

```
##   ID Genre Age
## 1  1 Homme  45
## 5  5 Homme  44
```

Il est également possible de mettre plusieurs conditions.

```
# Extraction de ligne
dataframe[dataframe$Genre == "Femme" & dataframe$Age < 44, ]
```

```
##   ID Genre Age
## 2  2 Femme  42
## 4  4 Femme  43
```

3.2 Opérateurs logiques

Opérateur	Description
<	strictement inférieur
<=	inférieur ou égal
>	strictement supérieur
>=	supérieur ou égal

Opérateur	Description
==	égal
!=	différent
!x	non x
x | y	x ou y
x & y	x et y

Chapter 4

Gestion des données

Plusieurs fonctions permettent d'obtenir des informations concernant la base de données. Le code suivant vous propose plusieurs fonction avec en commentaire leur utilités.

```
str(iris) #montre la structure de la base de données
```

```
## 'data.frame':    150 obs. of  5 variables:
##  $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
##  $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
##  $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
##  $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
##  $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...
```

```
dim(iris) #donnes les dimensions de la base de données
```

```
## [1] 150   5
```

```
nrow(iris) #donne le nombre de lignes de la base de données
```

```
## [1] 150
```

```
ncol(iris) #donne le nombre de colonnes de la base de données
```

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

```
head(iris) #affiche les premières lignes de la base de données, par défaut 5 lignes
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
## 4          4.6         3.1          1.5         0.2  setosa
## 5          5.0         3.6          1.4         0.2  setosa
```

```
## 6          5.4         3.9          1.7         0.4  setosa
```

```
tail(iris, n = 10) #affiche les dix dernières lignes de la base de données
```

```
##     Sepal.Length Sepal.Width Petal.Length Petal.Width   Species
## 141          6.7         3.1          5.6         2.4 virginica
## 142          6.9         3.1          5.1         2.3 virginica
## 143          5.8         2.7          5.1         1.9 virginica
## 144          6.8         3.2          5.9         2.3 virginica
## 145          6.7         3.3          5.7         2.5 virginica
## 146          6.7         3.0          5.2         2.3 virginica
## 147          6.3         2.5          5.0         1.9 virginica
## 148          6.5         3.0          5.2         2.0 virginica
## 149          6.2         3.4          5.4         2.3 virginica
## 150          5.9         3.0          5.1         1.8 virginica
```

```
summary(iris) #donne les résumés numériques des variables en fonction de leur type
```

```
##   Sepal.Length    Sepal.Width     Petal.Length    Petal.Width
##  Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100
##  1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300
##  Median :5.800   Median :3.000   Median :4.350   Median :1.300
##  Mean   :5.843   Mean   :3.057   Mean   :3.758   Mean   :1.199
##  3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100   3rd Qu.:1.800
##  Max.   :7.900   Max.   :4.400   Max.   :6.900   Max.   :2.500
##        Species
##  setosa    :50
##  versicolor:50
##  virginica :50
##
##
##
```

Chapter 5

Packages et données

Ce premier chapitre introduit deux concepts importants dans R. Le premier est les packages et le second concerne les données.

5.1 Installation et gestion des packages

Les packages sont des regroupements de fonctions et de jeux de données développés dans doivent se télécharger une seule fois, mais ils devront être importés à chaque utilisation. Le code ci-dessous permet d'installer le package ggplot2:

```
install.packages(ggplot2, dependencies = TRUE)
```

Avant chaque utilisation des packages, il est nécessaire d'importer le package grâce au code suivant:

```
library(ggplot2)
```

5.2 Téléchargement des données

Dans R, les bases de données se déclinent de plusieurs façons:

1. Les bases de données peuvent être directement incluses dans R ou dans les packages.
2. Les bases de données peuvent être créées dans l'environnement sauvées dans l'environnement R. Ces fichiers ont une extension .RData
3. Les bases de données peuvent être issues de fichiers externes. Ces fichiers peuvent avoir différentes extensions, les plus courantes étant .csv et .txt.

5.2.1 Bases de données issues de la base de R ou des packages

Le code suivant permet d'importer le jeu de données "iris" disponible dans R.

```
data("iris")
```

Si les données sont dans un package, le package doit être importé au préalable.