

Programmation sous R

Chapitre 1: Initiation au logiciel R

Mohamed Essaied Hamrita
mhamrita@gmail.com
github.com/Hamrita

Université de Sousse - Tunisie

2023-2024

- 1 Introduction
- 2 Démarrer avec R
- 3 Les différents objets
- 4 Fonctions utiles
- 5 Traitement des données

Section 1

Introduction

Introduction

R est un logiciel de calcul scientifique interactif **libre** et **gratuit** qui possède une large collection d'outils statistiques et graphiques. Plusieurs sites sont consacrés à ce logiciel. Parmi lesquels, je cite:

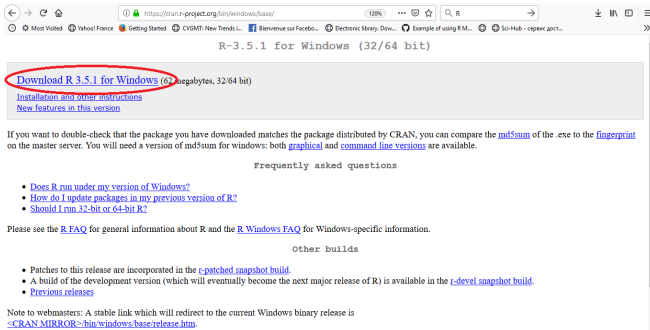
- <http://http://www.r-project.org/>. Le site officiel du logiciel dans lequel on trouve une description exhaustive sur le langage R et fournit les liens indispensables pour les différents téléchargements.
- <http://www.statmethods.net/>. QuickR, un site dans lequel on trouve les fonctions les plus utiles lors d'une analyse statistique uni et multi-variée.
- <https://www.learnbyexample.org/r/> Apprendre R par des exemples.

Installation du logiciel R

Dans la page de r-project, on trouve des versions de R compilée et sont disponibles pour Windows, Linux et Mac OS X. Ici, on décrit l'installation de R sous Windows. Tout d'abord, on se rendre sur cette page: <http://cran.r-project.org/bin/windows/base/> et l'on suivra le premier lien pour télécharger le programme d'installation.

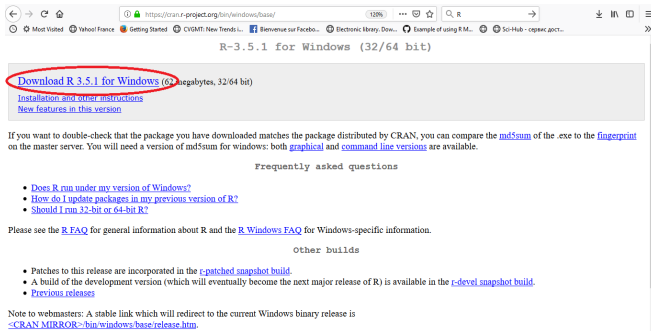
Installation du logiciel R

Dans la page de r-project, on trouve des versions de R compilée et sont disponibles pour Windows, Linux et Mac OS X. Ici, on décrit l'installation de R sous Windows. Tout d'abord, on se rend sur cette page: <http://cran.r-project.org/bin/windows/base/> et l'on suivra le premier lien pour télécharger le programme d'installation.



Installation du logiciel R

Dans la page de r-project, on trouve des versions de R compilée et sont disponibles pour Windows, Linux et Mac OS X. Ici, on décrit l'installation de R sous Windows. Tout d'abord, on se rend sur cette page: <http://cran.r-project.org/bin/windows/base/> et l'on suivra le premier lien pour télécharger le programme d'installation.



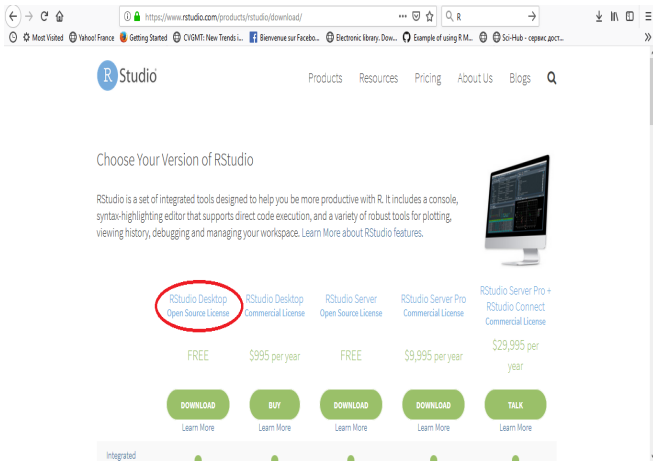
Une fois le programme d'installation lancé, il suffira d'installer R avec les options par défaut.

Installation de RStudio

Une fois R est correctement installé, je vous recommande fortement d'installer RStudio qui est téléchargeable depuis <http://www.rstudio.com/products/rstudio/download/>:

Installation de RStudio

Une fois R est correctement installé, je vous recommande fortement d'installer RStudio qui est téléchargeable depuis <http://www.rstudio.com/products/rstudio/download/>:



The screenshot shows the RStudio website's download page. The browser address bar displays the URL <https://www.rstudio.com/products/rstudio/download/>. The page features the RStudio logo and navigation links: Products, Resources, Pricing, About Us, and Blogs. The main heading is "Choose Your Version of RStudio". Below this, a paragraph describes RStudio as a set of integrated tools for R, including a console, editor, and plotting tools. To the right is an image of a computer monitor displaying the RStudio interface. The page lists five product options in a table-like format:

Product	License	Price	Action	Learn More
RStudio Desktop	Open Source License	FREE	DOWNLOAD	Learn More
RStudio Desktop	Commercial License	\$995 per year	BUY	Learn More
RStudio Server	Open Source License	FREE	DOWNLOAD	Learn More
RStudio Server Pro	Commercial License	\$9,995 per year	DOWNLOAD	Learn More
RStudio Server Pro + RStudio Connect	Commercial License	\$29,995 per year	TALK	Learn More

At the bottom, there is a section labeled "Integrated" with a progress bar showing the integration status of each product.

Mise à jour de R

Pour mettre à jour R sous Windows, il suffit de télécharger et installer la dernière version du programme d'installation.

Il est à remarquer que la nouvelle version sera installée à côté de l'ancienne version. Afin de libérer de l'espace, vous pouvez désinstaller l'ancienne version en utilisant l'utilitaire **désinstaller un programme** de Windows.

Lorsque plusieurs versions de R sont disponibles, RStudio choisit par défaut la plus récente.

Section 2

Démarrer avec R

Le répertoire courant

Pour pouvoir récupérer des données, il est utile de connaître le répertoire de travail, c'est-à-dire le répertoire sous lequel les divers résultats seront sauvegardés par défaut. Ce dernier s'obtient à l'aide la commande :

```
getwd()
```

```
[1] "D:/Enseignement/R/Diapos_2023/R/Chap1"
```

Tandis que la commande

```
setwd("C:/User/Mes documents/CoursR")
```

change de répertoire courant.

Démarrer avec R

```
# ceci est un commentaire
```

```
# opérateurs: +, -, *, ^ (**), %/%, %%
```

```
# opérations matricielles: %*%, solve(), as.vector(),
```

```
# det(), t()
```

```
5%/3
```

```
[1] 1
```

```
cos(pi/3)
```

```
[1] 0.5
```

```
## Saisie des données
```

```
XX <- c("M", "M", "D", "C", "C", "M", rep("C", 3), "M", "C",  
       "M", "V", "M", "V", "D", rep("C", 3), "M")
```

```
# Créer un objet x en lui affectant le nombre 2  
x <- 2  
nom <- "Mohamed"  
# Créer un objet x et afficher son contenu  
(x <- 2)
```

```
[1] 2
```

Le logiciel R travail avec des objets. Mais quelle est la classe de ces objets? de quels modes?

Section 3

Les différents objets

Les différents objets

Les différents objets de R sont: vecteur, matrice, liste, tableau des données ou ts (time series).

- Vecteur: un vecteur peut être de mode numérique, caractère, complexe ou logique.

```
x1<-c(1,-5)  # vecteur numérique  
mode(x1)     # afficher le mode de l'objet x1.
```

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

```
x2<-c("Mohamed","Sarah") # vecteur caractère  
x3<-c(1i,1-1i,-2+3i) # vecteur complexe  
x4<-c(TRUE,FALSE,T,F)  # vecteur logique
```

Pour créer des séries régulières, on peut utiliser les fonctions suivantes:

```
# c(), seq(), : ou rep()
```


La fonction c()

Pour avoir l'aide de cette fonction, tapez ?c

```
x1<-c(1,0,5,-4)  # création d'un vecteur
x1[3]            # extraction du troisième élément de x1
```

```
[1] 5
```

```
x1[-1]          # afficher tous les éléments de x1 sauf le premier
```

```
[1] 0 5 -4
```

```
x1[x1>2]        # extraire les éléments supérieur à 2.
```

```
[1] 5
```

```
x1>2            # vecteur logique pour tester si x1>2 ou non.
```

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

```
x11<-c("a","A","b","B")
lettres15<-letters[c(1,5)] # création d'un vecteur contenant la
# première et la cinquième lettres minuscules
lettres15
```

```
[1] "a" "e"
```

```
LETTRES15<-LETTERS[c(1,5)] # idem mais majuscules
LETTRES15
```

```
[1] "A" "E"
```

La fonction seq()

la fonction seq() est utilisée pour créer des séquences.

```
seq1<-seq(0,1,by=0.1)  # séquence de 0 à 1 par incrémentation 0.1.  
seq1
```

```
[1] 0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0
```

```
length(seq1)  # afficher la longueur de seq1
```

```
[1] 11
```

```
seq2<-seq(0,1,len=5)  # séquence de 0 à 1 de longueur 5.  
seq2
```

```
[1] 0.00 0.25 0.50 0.75 1.00
```

```
seq3<-seq(0,1,len=11) # même résultat que seq1  
seq3
```

```
[1] 0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0
```

Les fonctions : et rep()

la fonction : est utilisée pour créer des séquences entière de a à b.

```
seq<-1:5 # séquence entière de 1 à 5.  
seq
```

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

```
seqi<-6:-4  
seqi
```

```
[1] 6 5 4 3 2 1 0 -1 -2 -3 -4
```

La fonction rep() est utilisée lorsqu'on veut répéter un élément ou un vecteur n fois.

```
rep(1,5)
```

```
[1] 1 1 1 1 1
```

```
rep(1:3,3)
```

```
[1] 1 2 3 1 2 3 1 2 3
```

```
rep(1:3,each=3)
```

```
[1] 1 1 1 2 2 2 3 3 3
```

```
rep(1:3,each=3,times=2)
```

```
[1] 1 1 1 2 2 2 3 3 3 1 1 1 2 2 2 3 3 3
```

```
rep(1:3,each=2,len=12)
```

```
[1] 1 1 2 2 3 3 1 1 2 2 3 3
```

```
rep(c(0,1,6),times=c(2,5,4))
```

```
[1] 0 0 1 1 1 1 1 6 6 6 6
```

```
rep("a",5)
```

```
[1] "a" "a" "a" "a" "a"
```

```
rep(1:3,each=2,len=12)
```

```
[1] 1 1 2 2 3 3 1 1 2 2 3 3
```

```
rep(c(0,1,6),times=c(2,5,4))
```

```
[1] 0 0 1 1 1 1 1 1 6 6 6 6
```

```
rep("a",5)
```

```
[1] "a" "a" "a" "a" "a"
```

- Les matrices

Le deuxième objet traité avec R est l'objet matrice (`matrix`). Pour créer un tel objet, on utilise la fonction `matrix()`

```
matrix(seq(-2,2,len=6),nrow=2,ncol=3)
```

```
      [,1] [,2] [,3]  
[1,] -2.0 -0.4  1.2  
[2,] -1.2  0.4  2.0
```

*# la fonction `matrix` remplit la matrice à créer colonne par colonne.
Pour faire le remplissage ligne par ligne, on ajoute l'argument `byrow=T`.*

```
matrix(seq(-2,2,len=6),nrow=2,ncol=3,byrow=T)
```

```
      [,1] [,2] [,3]  
[1,] -2.0 -1.2 -0.4  
[2,]  0.4  1.2  2.0
```

```
# création des matrices en utilisant cbind() et rbind().
```

```
age<-c(22,21,24)
poids<-c(64,56,70)
cbind(age,poids)
```

```
      age poids
[1,]  22    64
[2,]  21    56
[3,]  24    70
```

```
rbind(age,poids)
```

```
      [,1] [,2] [,3]
age      22  21  24
poids    64  56  70
```

```
M<-cbind(age,poids)
N<-rbind(age,poids)
colnames(M) # les noms des colonnes
```

```
[1] "age"  "poids"
```

```
rownames(N) # les noms des lignes
```

```
[1] "age"  "poids"
```

Quelques opérations sur les matrices

```
m1<-matrix(seq(-2,2,len=6),nrow=3,ncol=2)
m1
```

```
      [,1] [,2]
[1,] -2.0  0.4
[2,] -1.2  1.2
[3,] -0.4  2.0
```

```
t(m1)  # la transposée de m1
```

```
      [,1] [,2] [,3]
[1,] -2.0 -1.2 -0.4
[2,]  0.4  1.2  2.0
```

```
t(m1)%*%m1 # multiplication matricielle de m1' par m1
```

```
      [,1] [,2]
[1,]  5.60 -3.04
[2,] -3.04  5.60
```

```
det(t(m1)%*%m1) # le déterminant
```

```
[1] 22.1184
```

```
solve(t(m1)%*%m1)  # l'inverse.
```

```
      [,1]      [,2]  
[1,] 0.2531829 0.1374421  
[2,] 0.1374421 0.2531829
```

```
diag(t(m1)%*%m1)  # la diagonale
```

```
[1] 5.6 5.6
```

```
diag(c(1,-2,5))  # matrice diagonale
```

```
      [,1] [,2] [,3]  
[1,]    1    0    0  
[2,]    0   -2    0  
[3,]    0    0    5
```

```
sum(diag((t(m1)%*%m1)))  # la trace d'une matrice
```

```
[1] 11.2
```


- Les listes

La liste est le mode de stockage le plus général et polyvalent du langage R. Il s'agit d'un type de vecteur spécial dont les éléments peuvent être de n'importe quel mode

```
(list1 <- list(size = c(1, 5, 2), user = "Mohamed", new = TRUE))
```

```
$size
```

```
[1] 1 5 2
```

```
$user
```

```
[1] "Mohamed"
```

```
$new
```

```
[1] TRUE
```

```
list1[[1]]    # accéder au premier élément de list1
```

```
[1] 1 5 2
```

```
list1[["size"]] # idem ou encore, list1$size.
```

```
[1] 1 5 2
```

Section 4

Fonctions utiles

Arrondissement

On donne ici, quelques fonctions utiles.

```
# arrondissement à l'entier  
xx=c(0.253,2.146,pi,2*pi/3,11.5)  
round(xx)      # arrondi à l'entier
```

```
[1] 0 2 3 2 12
```

```
round(xx,2)    # arrondi à la seconde décimale
```

```
[1] 0.25 2.15 3.14 2.09 11.50
```

```
round(xx,-1)   # arrondi aux dizaines
```

```
[1] 0 0 0 0 10
```

```
ceiling(xx)    # plus petit entier supérieur
```

```
[1] 1 3 4 3 12
```

```
floor(xx)      # plus grand entier inférieur
```

```
[1] 0 2 3 2 11
```

```
trunc(xx)     # troncature des décimales
```

```
[1] 0 2 3 2 11
```

La fonction `apply()` et dérivées:

- La fonction `apply()` permet d'appliquer une fonction `FUN` (par exemple une moyenne, une somme) à chaque ligne (si `MARGIN=1`) ou à chaque colonne (si `MARGIN=2`) d'un tableau de données `x`.

La fonction `apply()` et dérivées:

- La fonction `apply()` permet d'appliquer une fonction `FUN` (par exemple une moyenne, une somme) à chaque ligne (si `MARGIN=1`) ou à chaque colonne (si `MARGIN=2`) d'un tableau de données `x`.
- La fonction `sapply(x, FUN)` permet d'appliquer la fonction `FUN` à tous les éléments de la liste (du vecteur) `x`.
- La fonction `outer(x,y, FUN)`: Retourner une matrice de la form

$$M_{ij} = FUN(x_i, y_j)$$

```
outer(X=c(1,-2,3),Y=c(3,2,-4,5),"+")
```

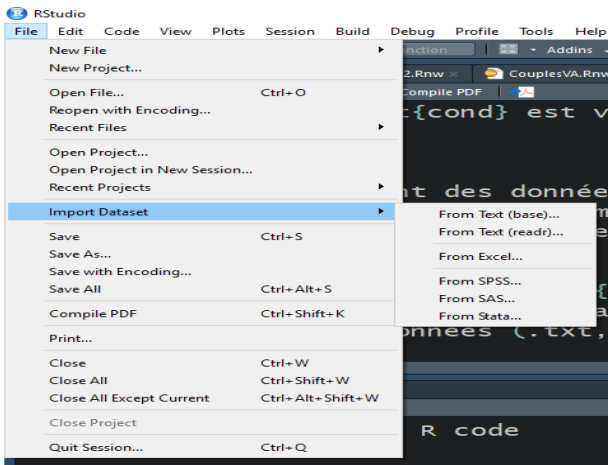
	[,1]	[,2]	[,3]	[,4]
[1,]	4	3	-3	6
[2,]	1	0	-6	3
[3,]	6	5	-1	8

Section 5

Traitement des données

Importation des données:

Avec R, il existe des diverses alternatives pour importer une base de données. On donne ici la méthode la plus simple. Il suffit de cliquer sur le bouton File du menu de RStudio, puis sur Import Dataset, ensuite choisir l'extension des données (.txt, .spss, etc).



Saisir des données au clavier:

En utilisant la fonction `scan()`, la saisie d'une série de données peut paraître moins fastidieuse.

```
jeu1=scan()  
1: -2  
2: 1  
3: 5  
4:  
jeu1
```

Le premier retour après une chaîne vide met fin à la saisie

Exporter une base de données:

Pour exporter un tableau de données, on fait appel à la fonction `write.csv(x, file="", ...)` où `x` est l'objet à exporter et `file` est une chaîne de caractère précisant l'emplacement où on veut enregistrer l'objet `x`.

```
xx=matrix(seq(0,5,len=200),nc=4)
write.csv(xx, "D:/Cours_R/tab.csv")
# enregistrer l'objet xx dans le répertoire D:/Cours_R
# sous le nom tab avec l'extension .csv
```