Initiation to the software

Contents

Extract from Wikipedia :

« R is a programming language and software environment for statistical computing and graphics supported by the R Foundation for Statistical Computing. The R language is widely used among statisticians and data miners for developing statistical software and data analysis. Polls, surveys of data miners, and studies of scholarly literature databases show that R's popularity has increased substantially in recent years.

R is a GNU package. The source code for the R software environment is written primarily in C, Fortran, and R.R is freely available under the GNU General Public License, and pre-compiled binary versions are provided for various operating systems. While R has a command line interface, there are several graphical front-ends available.»

1. Download and install R

1.1 The software R-base

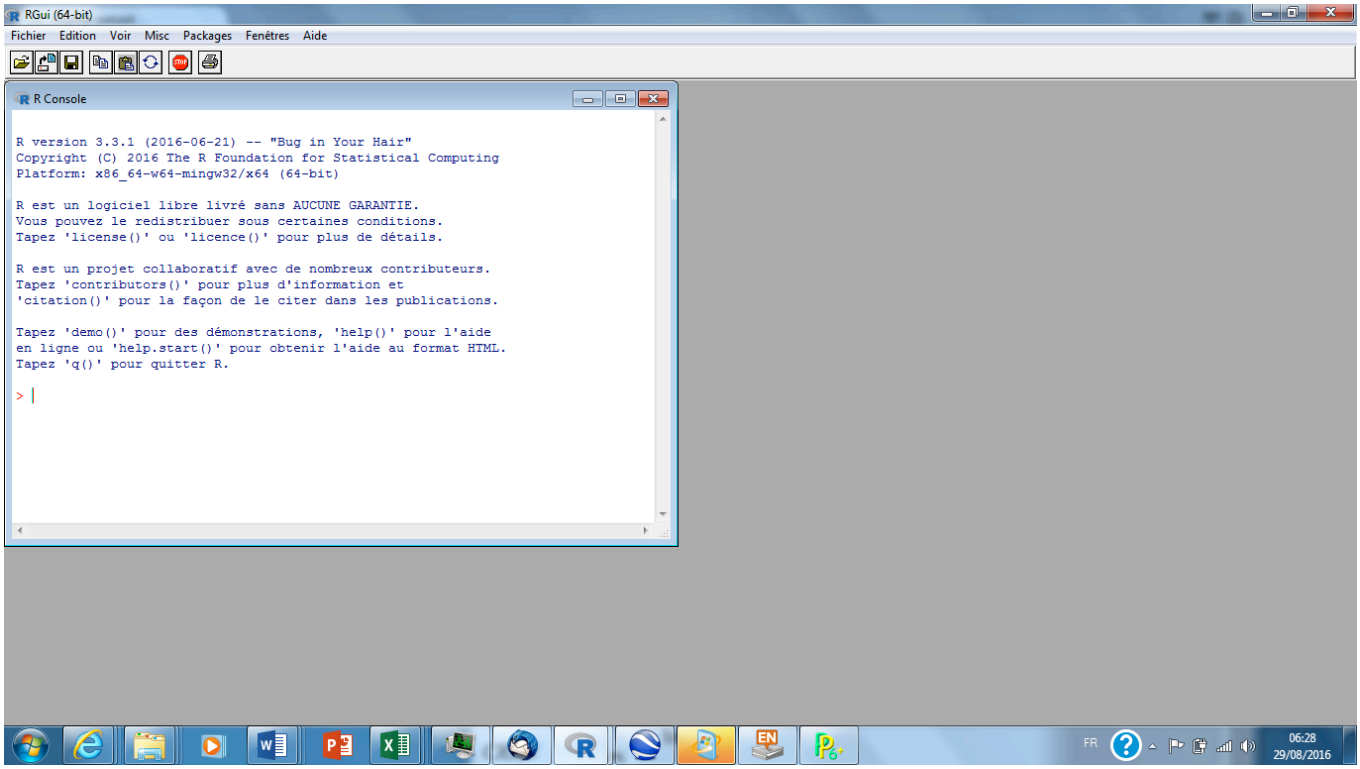
 is composed of 2 elements : **(1) the software**, **(2) the packages in addition** that have to be separately installed and will give further functionalities.

Web site to download (software + packages), help, forum…

http://cran.r-project.org/

Installation

1. Go to : http://cran.r-project.org/
2. Go to the section *Download and Install R* and choose an OS
3. *base/ Install R for the first time*
4. *Download R 3.3.1 for Windows*
6. Double click on the file and follow the instructions
7. Then click on  . A window called "**fenêtre" is the R console**, with > (in red). This symbol is « Invite de commande », tell us where to write the instructions.



1.2. R studio

Studio is an interface to facilitate the use of .

You will first need to download

Installation

1. Go to : **https://www.rstudio.com/Download Rstudio**
2. Click on Download (e.g. *RStudio 0.99.903 - Windows Vista/7/8/10*)
4. Follow the instructions
7. Run the software by clicking on Studio. You will get 3 windows. ON the left the console, with the symbol > (in red): "Invite de commande ". On the top right, you will find the window « Environment » where will be listed the data with their features. The second window will help to display your files, graphics, packages and the help.

Specify the working directory by default

1.3. Packages

To carry out specific tasks. They contain dedicated functions for this specific task.

1.3.1. To install a software

►ON ,

1. Click on *Packages*, on the left (*Software*). Have a look to '*Available*. Click on it to get a short presentation of the package.
2. Slect the mirror *CRAN* from the panel *Packages*
3. Select *Installer le(s) Package*

►Pour RStudio ,

1. Click on *Tools/install packages*

3. Give the name of the package

1.3.2. Load into memory the package by using the library function

```
>library(car)
```

You will have to repeat this operation each time you run R.

2. How to work with R and RStudio ?

2.1. The « console » window

In RStudio, the window console R is on the left. It is where you will write your instructions or code.

-> What you type will be in red, the answer or result will be in blue.

```
>2+2
```

2.2. The right working method

2.2.1. Set the working directory

►For RStudio , you have to select your working directory : *Tools / Global Options* and set the path in *Default working directory*

2.2.2. Start to work using a script

They are useful to create a set of instructions in R (as functions) and to keep them into memory for a later use. Thus, you can access to an older script previously saved .

► RStudio, select **`File / New file / R script`** to create a new file or **`File / Open file (ou Ctrl+O)`** to re-open an older one.

2.3. How to save with R

►Pour R ,

-> **The instructions you used are saved in your working directory with the file .Rhistory**

-> **the session** : contains the objects you crated. ***ls()*** to get them

2.4. Simple use of R

2.4.1. The calculator

All the classical operations can be used +, -, *, /, exp(), log(). Try log(10) then enter.

```
> log(10)
[1] 2.302585
```

2.4.2. to clean up the console

Edit/Clear console (« Crtl+L)

2.4.3. Functions

The name of the function with the parameters between ()

The simplest function is *c()* to create a vector (series of numbers) with several values.

```
> c(15,18,23,19,24)
[1] 15 18 23 19 24
```

⚠ You have to follow the syntaxes of the functions

2.4.4. Assignation

To save the result of a function, you have to assign it to a variable.

```
> Temp<-c(15,18,23,19,24)
```

The object Temp will appear in the top right window.

Warning!!! Temp is different of *TEMP*

```
> temp
Erreur : objet 'temp' introuvable (not found)
> Temp
[1] 15 18 23 19 24
```

The vector is saved in memory :

```
> (Temp*9+160)/5
[1] 59.0 64.4 73.4 66.2 75.2
```

```
> mean(Temp)
[1] 19.8
```

2.4.5. To create a vector/list of numbers

a : b to get the numbers between a and b

```
> 4:19
 [1]  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19
```

seq() to do the same but with a start and end and a specific jump (here : 2)

```
> seq(from = 0, to = 12, by = 2)
[1]  0  2  4  6  8 10 12
```

rep()

```
> rep(c(15,18,23), each = 3, times = 2)
 [1] 15 15 15 18 18 18 23 23 23 15 15 15 18 18 18 23 23 23
```

2.4.6. A matrix

2.4.6.1. numerical data

matrix() A **matrix** is a collection of data elements arranged in a two-dimensional rectangular layout. The following is an example of a matrix with 5 rows and 5 columns.

```
> matrice <- matrix(1:25, ncol = 5, byrow=FALSE)
> matrice
   [,1] [,2] [,3] [,4] [,5]
[1,]   1   6  11  16  21
[2,]   2   7  12  17  22
[3,]   3   8  13  18  23
[4,]   4   9  14  19  24
[5,]   5  10  15  20  25
```

Give a name to rows and columns

```
> colnames(matrice) <- c("A","B","C","D","E")
> rownames(matrice) <- c(1:5); matrice
  A  B  C  D  E
1 1  6 11 16 21
2 2  7 12 17 22
3 3  8 13 18 23
4 4  9 14 19 24
5 5 10 15 20 25
```

to display a value of interest

```
> matrice[2,3]
```

To display a row or a column, *for instance the row 2, then a comma and nothing is specified for the column (to get the complete row).*

```
> matrice[2,]
[1]  2  7 12 17 22
```

To display a sub-part of the matrix using c(). From matrice, you will get the rows 2 and 5, with the columns 1,3 and 4.

```
> matrice[c(2,5), c(1,3,4)]
     [,1] [,2] [,3]
[1,]    2   12   17
[2,]    5   15   20
```

To replace a value in the matrix

```
> matrice[4,5]<-40 ; matrice
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    6   11   16   21
[2,]    2    7   12   17   22
[3,]    3    8   13   18   23
[4,]    4    9   14   19   40
[5,]    5   10   15   20   25
```

To delete a value or a complete row.

```
> matrice[-2,]
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    6   11   16   21
[2,]    3    8   13   18   23
[3,]    4    9   14   19   40
[4,]    5   10   15   20   25
```

To carry out a calculus with a complete line/column in a matrix, for example to get the mean of column .

```
> mean(matrice[-2,4])
[1] 18.25
```

To select some values according to a condition (with >,>=,==…)

```
 > matrice[matrice<=5]
[1] 1 2 3 4 5
> matrice[matrice<15 & matrice>=7]
[1]  7  8  9 10 11 12 13 14
```

->**To replace the value** ≤ *5 with 0***.**

```
> matrice[matrice<=5]<-0
> matrice
     [,1] [,2] [,3] [,4] [,5]
[1,]    0    6   11   16   21
[2,]    0    7   12   17   22
[3,]    0    8   13   18   23
[4,]    0    9   14   19   40
[5,]    0   10   15   20   25
```

2.4.6.2. Numerical data or/and qualitative. To mix them in the same object

We use the function `data.frame()`. String will be given with “”..

```
>qqLettres=data.frame(lettre=c("A","b","F","e"),ordre=c(1,2,6,
5),casse=c("maj","min","maj","min"))
>qqLettres
  lettre ordre casse
  1      A     1   maj
  2      b     2   min
  3      F     6   maj
  4      e     5   min
```

 The manipulation of a `data.frame` is similar to the manipulation of a `matrix`.

3. The help in R ?

3.1. To start or to improve your R programming

Have a look to http://www.r-project.org/, to see the Documentations in Manuals then Contributed.

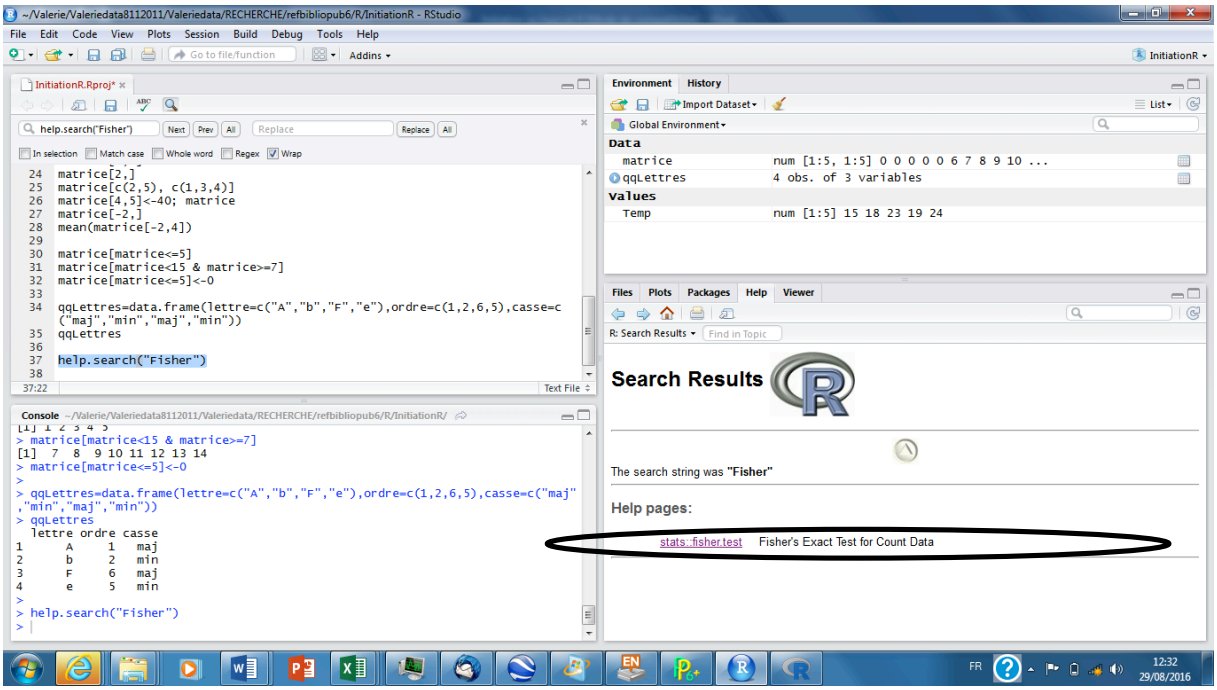
Ex: “R for Beginners", “Fitting Distributions with R", “Practical Regression and Anova using R”…

3.2. To carry out a specific test or build a graphic

3.2.1. If you have no idea of the name of the function

Use *help.search()* with the name of test between “”

```
> help.search("Fisher")
```



OR Ask for help through Rstudio

```
> help.start()
```

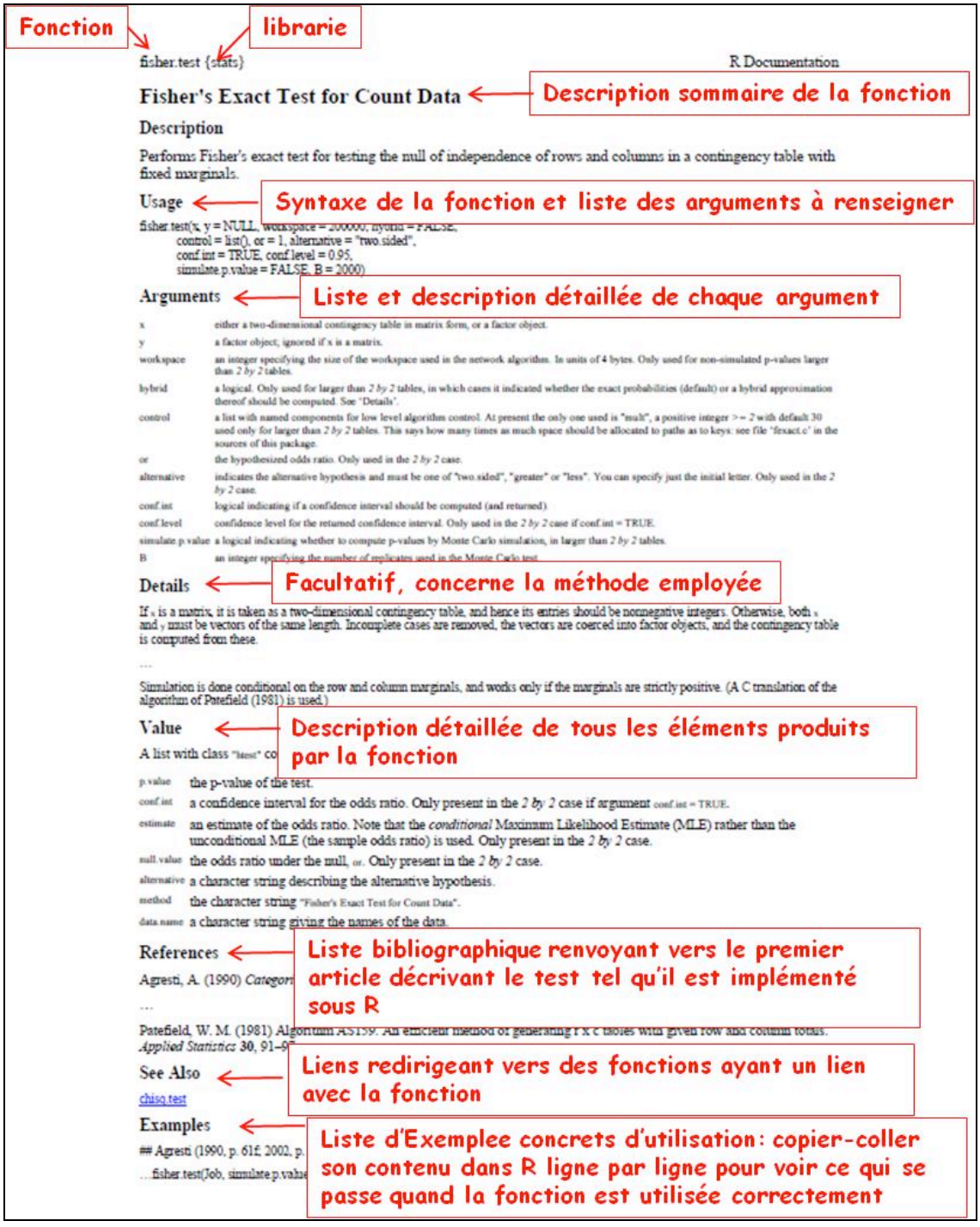
Then click on *Search Engine & Keywords* .

3.2.3. You know the name of the function you want to use

Start with ? with the name of the function

```
> ?fisher.test()
```

La fenêtre d'aide s'affiche en bas à droite contenant toutes les informations nécessaires à l'utilisation de la fonction. Tous les fichiers d'aide sont construits de la même manière :



⚠ The section *Usage* and *Examples* are the 2 important sections
In the usage section, you will get the syntax to use the function.

Usage

```
fisher.test(x, y = NULL, workspace = 200000, hybrid = FALSE,
        control = list(), or = 1, alternative = "two.sided",
        conf.int = TRUE, conf.level = 0.95,
        simulate.p.value = FALSE, B = 2000)
```

Each argument is separated by a comma. Only few of them are mandatory. Here : *x*. The others arguments are followed by '= something' with default values et then are optionals. You will change them if you need to.

⚠ You are not obliged to give their names only if you use them n the same order.

3.3. If it is not enough…

Google with keywords, example « test Fisher R CRAN ». Many forum will give you pertinent information.

4. How to import a table in R?

This is possible with files coming from  or  .

4.1. The right format of your csv file

The basic rules for making use of the file (on moodle) `iristableur.xls`. This file contains information about 50 units of 3 iris species : *Iris setosa*, *diversicolor* et *virginica.*

- No space in the names *Exemple : Sepal.Length au lieu de Sepal Length.*
- Replace the missing data by NA
- A name for each lines
- Save the file as `irisdata.txt`

	Code.Iris	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
1	S1	5,1	3,5	1,4	0,2	setosa
2	S2	4,9	3	1,4	0,2	setosa
3	S3	4,7	3,2	1,3	0,2	setosa
4	S4	4,6	3,1	1,5	0,2	setosa
5	S5	5	3,6	1,4	0,2	setosa
		…	…	…	…	…
51	V1	7	3,2	4,7	1,4	versicolor
52	V2	6,4	3,2	4,5	1,5	versicolor
53	V3	6,9	3,1	4,9	1,5	versicolor
54	V4	5,5	2,3	4	1,3	versicolor
55	V5	6,5	2,8	4,6	1,5	versicolor
		…	…	…	…	…
101	VG1	6,3	3,3	6	2,5	virginica
102	VG2	5,8	2,7	5,1	1,9	virginica
103	VG3	7,1	3	5,9	2,1	virginica
104	VG4	6,3	2,9	5,6	1,8	virginica
105	VG5	6,5	3	5,8	2,2	virginica
		…	…	…	…	…

4.2. Importation du fichier sous R

Use the function **read.table()**. Here are the most important arguments to import `irisdata.txt` and to put it in `irisdata`

```
>irisdata<-read.table("irisdata.txt",header=TRUE,row.names=1,dec=",")
```

OR

```
>irisdata<-read.table("irisdata.csv",header=TRUE,row.names=1,dec=",")
```

file .txt for  or .csv for 

- The first argument has to be the name of the file between “”
- `header` is the names of the columns;
- `dec` the character used in the file for decimal points.
- `row.names` a vector of row names. This can be a vector giving the actual row names, or a single number giving the column of the table which contains the row names, or character string giving the name of the table column containing the row names.

  If there is a header and the first row contains one fewer field than the number of columns, the first column in the input is used for the row names. Otherwise if row.names is missing, the rows are numbered.
- The file will be stored in the object `irisdata` to make use of it

`irisdata` is a matrix of 150 lines and 5 columns.

- `names()` to get the names of the columns
- `head()` to get the first (6) lines of the file
- `dim()` to get the number of lines and columns of the file
- `View()` to get the complete file in the top window

```
> names(irisdata)
[1] "Sepal.Length" "Sepal.Width"  "Petal.Length" "Petal.Width"
"Species"
> head(irisdata)
        Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
  S1             5.1         3.5          1.4         0.2     setosa
  S2             4.9         3.0          1.4         0.2     setosa
  S3             4.7         3.2          1.3         0.2     setosa
  S4             4.6         3.1          1.5         0.2     setosa
  S5             5.0         3.6          1.4         0.2     setosa
  S6             5.4         3.9          1.7         0.4     setosa

> dim(irisdata)
 [1] 150   5
> View(irisdata)
```

5. How to manipulate a table

5.1. To display a variable (columns)

To get the third column of `irisdata`

```
> irisdata[,3]
  [1] 1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 1.5 1.6 1.4 1.1 1.2 1.5 1.3 1.4
 [19] 1.7 1.5 1.7 1.5 1.0 1.7 1.9 1.6 1.6 1.5 1.4 1.6 1.6 1.5 1.5 1.4 1.5 1.2
 [37] 1.3 1.4 1.3 1.5 1.3 1.3 1.3 1.6 1.9 1.4 1.6 1.4 1.5 1.4 4.7 4.5 4.9 4.0
 [55] 4.6 4.5 4.7 3.3 4.6 3.9 3.5 4.2 4.0 4.7 3.6 4.4 4.5 4.1 4.5 3.9 4.8 4.0
 [73] 4.9 4.7 4.3 4.4 4.8 5.0 4.5 3.5 3.8 3.7 3.9 5.1 4.5 4.5 4.7 4.4 4.1 4.0
 [91] 4.4 4.6 4.0 3.3 4.2 4.2 4.2 4.3 3.0 4.1 6.0 5.1 5.9 5.6 5.8 6.6 4.5 6.3
[109] 5.8 6.1 5.1 5.3 5.5 5.0 5.1 5.3 5.5 6.7 6.9 5.0 5.7 4.9 6.7 4.9 5.7 6.0
[127] 4.8 4.9 5.6 5.8 6.1 6.4 5.6 5.1 5.6 6.1 5.6 5.5 4.8 5.4 5.6 5.1 5.1 5.9
[145] 5.7 5.2 5.0 5.2 5.4 5.1
```

Using the name of a variable

```
> irisdata$Petal.Length
  [1] 1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 1.5 1.6 1.4 1.1 1.2 1.5 1.3 1.4
[145] 5.7 5.2 5.0 5.2 5.4 5.1
```

Using *attach() or detach().*

```
> attach(irisdata)
> Petal.Length
  [1] 1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 1.5 1.6 1.4 1.1 1.2 1.5 1.3 1.4
[145] 5.7 5.2 5.0 5.2 5.4 5.1
> Sepal.Width
[1] 3.5 3.0 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 3.7 3.4 3.0 3.0 4.0 4.4 3.9 3.5
[145] 3.3 3.0 2.5 3.0 3.4 3.0
```

5.2. To change the name of lines or columns

colnames()

```
> colnames(irisdata)=c('Lsepale','larsepale','Lpetale',
'larpetale', 'Especes');head(irisdata)
   Lsepale larsepale Lpetale larpetale Especes
s1     5.1       3.5     1.4       0.2  setosa
s2     4.9       3.0     1.4       0.2  setosa
s3     4.7       3.2     1.3       0.2  setosa
s4     4.6       3.1     1.5       0.2  setosa
s5     5.0       3.6     1.4       0.2  setosa
s6     5.4       3.9     1.7       0.4  setosa
```

5.3. To remove lines with NA (missing data)

na.omit()

```
> irisdata<-na.omit(irisdata) ; dim(irisdata)
```

Only 139 lines are now in your object

5.4. To select elemnts according to conditions

• **To select samples according their sepal length** >= 6,5

```
> subset(irisdata&Lpetale>=6.5)
          Lsepale larsepale    Lpetale larpetale  Especes
VG6           7.6         3.0         6.6         2.1 virginica
VG18          7.7         3.8         6.7         2.2 virginica
VG19          7.7         2.6         6.9         2.3 virginica
VG23          7.7         2.8         6.7         2.0 virginica
```

• **To select samples according their sepal length** >= 6,5 and their sepal width < 3

```
> subset(irisdata&Lpetale>=6.5 & larsepale <3)
          Lsepale larsepale    Lpetale larpetale  Especes
VG19          7.7         2.6         6.9         2.3 virginica
VG23          7.7         2.8         6.7         2.0 virginica
```

• **To select data for the specie** *I. setosa*

```
> subset(irisdata&Especes=="setosa")
    Lsepale larsepale Lpetale larpetale Especes
S1      5.1       3.5     1.4       0.2  setosa
S2      4.9       3.0     1.4       0.2  setosa
S3      4.7       3.2     1.3       0.2  setosa
S4      4.6       3.1     1.5       0.2  setosa
…
```

• **To only select 2 variables for the specie** *I. setosa*

```
> subset(irisdata&Especes=="setosa",select=c('Lsepale','Lpetale'))
    Lsepale Lpetale
S1      5.1     1.4
S2      4.9     1.4
S3      4.7     1.3
…
```

5.5. To sort a table according a condition

• `order()`: increasing order according to a column

```
> irisdata[order(irisdata$Lpetale),]
         Lsepale larsepale    Lpetale larpetale  Especes
s23          4.6        3.6         1.0        0.2     setosa
s14          4.3        3.0         1.1        0.1     setosa
s15          5.8        4.0         1.2        0.2     setosa
… ….
```

• `rev()`: decreasing order

```
> irisdata[rev(order(irisdata$Lpetale)),]
    Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
vg19          7.7         2.6          6.9         2.3  virginica
vg23          7.7         2.8          6.7         2.0  virginica
vg18          7.7         3.8          6.7         2.2  virginica
…
```

5.6. To merge tables of data

• **To merge 2 or more tables:** `rbind()`

⚠ *They must have the same number of columns*

```
>irisdata[irisdata$Especes=="setosa",c(1,2)]->setosa
>irisdata[irisdata$Especes=="virginica",c(1,2)]->virginica
>rbind(setosa,virginica)
```

• **To merge 2 or more tables side by side:** `cbind()`

⚠ *They must have the same number of lines*

```
>irisdata[irisdata$Especes=="setosa",c(1,2)]->sepal
>irisdata[irisdata$Especes=="setosa",c(3,4)]->petal
>cbind(sepal,petal)->setosa
```

5.7. To import results under

The function `write.table()` help to export data for example

```
> write.table(setosa,"setosa.txt",row.names=FALSE,sep="\t")
```

- Give the filename
- `row.names=FALSE` if you do not want to export the names of the lines
- `sep` to design the separator (tabulation here)

Open the file with .

5.8. Simple statistics

• Simple statistics for each variables of the table

```
> summary(irisdata)
 Lsepale        larsepale        Lpetale        larpetale
Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100
1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.550   1st Qu.:0.300
Median :5.800   Median :3.000   Median :4.200   Median :1.300
Mean   :5.812   Mean   :3.059   Mean   :3.709   Mean   :1.176
3rd Qu.:6.400   3rd Qu.:3.350   3rd Qu.:5.100   3rd Qu.:1.800
Max.   :7.900   Max.   :4.200   Max.   :6.900   Max.   :2.500
      Especes
setosa    :48
versicolor:47
virginica :44
```

Note: For the columns with qualitative values (as the name of species) the number categories will be reported.

• The function for each variables `mean(), median(), min(), quantile(), max(), sd()`

```
>quantile(irisdata[,3], 0.25)
25%
1.55
> median(irisdata[,3])
[1] 4.2
> sd(irisdata[,3])
 [1] 1.774354
```

• Mean or Sum for each lines or columns. `colMeans()`, `rowMeans()`, `colSums()` and `rowSums()`.

```
> colMeans(setosa)
  Lsepale larsepale
 4.991667  3.406250
```

• To apply an operation (mean, standard deviation) **for a qualitative variable according to category:** function `tapply(X, INDEX, FUN)` where `X` is the variable for interest, `INDEX` qualitative variable (different categories) and `FUN` the function. For example, what is the mean by specie for the length of the sepals.

```
>tapply(X=irisdata$Lsepale, INDEX=irisdata$Especes,FUN=mean)
    setosa versicolor  virginica
  4.991667   5.942553   6.568182
```

• To centre and reduce the data: function `scale()`

```
>scale(setosa, center=TRUE, scale=TRUE)
```

6. How to build Graphics?

6.1. To display a graphic

Some functions **create a new window before building the graphic** (examples: `plot()` ; Other do not create a new graphic **but add new information to the graphic** (`axis()`, `title()`).

By default, a new graphic erase the older one. The function `windows()` (no arguments) or `x11 ()` is useful to open a new window without deleting the older one.

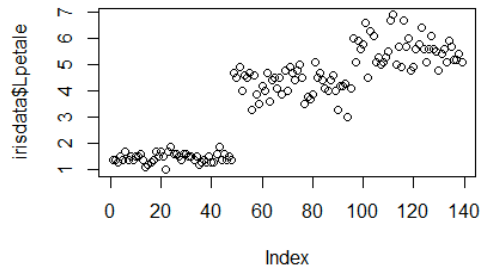
To get several graphics on the same window see the function `par(mfrow=c(nombre1,nombre2))` . « `nombre1` » to define the number of lines and « `nombre 2` », the number of columns. The number of graphics is equal to the product of this two numbers. *`par(mfrow=c(3,2))` wil diplay 6 graphics.*

6.2. *plot()*

`plot()` ***creates a graphical window, then display a graphic.***

```
> plot(irisdata$Lpetale)
```

To save it, see ***Export/Save as image***.



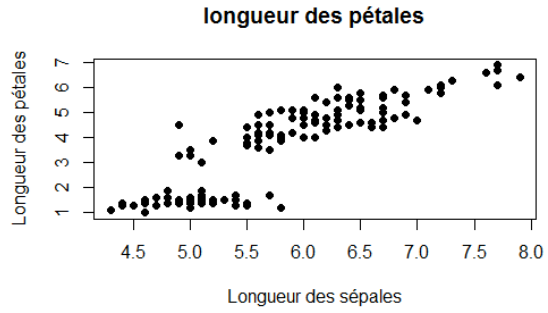
Use of *plot()*

```
> plot(varY~varX, type= « nom_type_trait* », pch= « »,col= «  »,
bg=«  », cex=, main= « », xlab =« »,ylab=« »,
xlim=c(nombre_min,nombre_max), ylim=c(nombre_min,nombre_max))
```

- `type` : points « p » ; « l » lines ; « b » points (cf. p 32)
- `col` : color
- `bg` : background color
- `cex` : size
- `pch` : symbol (cf. p 32)
- `main` : title at the top
- `xlab, ylab` : titles of the axis
- `xlim, ylim` : specifies the x-axis limits for the current axes

```
> plot(Lpetale~Lsepale,
irisdata, type= 'p', pch= 16,
main='longueur des pétales',
xlab= ' Longueur des sépales
```

```
', ylab= 'Longueur des
pétales')
```



6.2.1. How to add axis or titles?

Use the function `axis()` and `title()`.

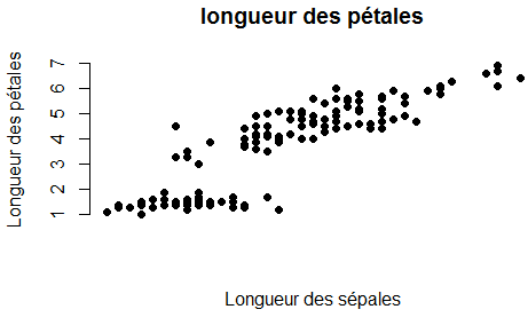
⚠ These functions do not create new graphics

- **to delete the axis rectangles** (the argument `axes=FALSE` in `plot()`)
- **To choose axis** (`axis(numero_axe)`). *To only keep the second axis: `axis(2)`*

Usages des fonctions axis() et title() :

```
> axis( chiffre1à4* , lty = '', lwd =, col ='',
col.axis= '')
> title (main ='', xlab = '', ylab = ''…)
```

```
> plot(Lpetale~Lsepale,
irisdata, type= 'p', pch= 16,
main='longueur des pétales',
xlab= ' Longueur des sépales
', ylab= 'Longueur des
pétales',axes=FALSE); axis(2)
```



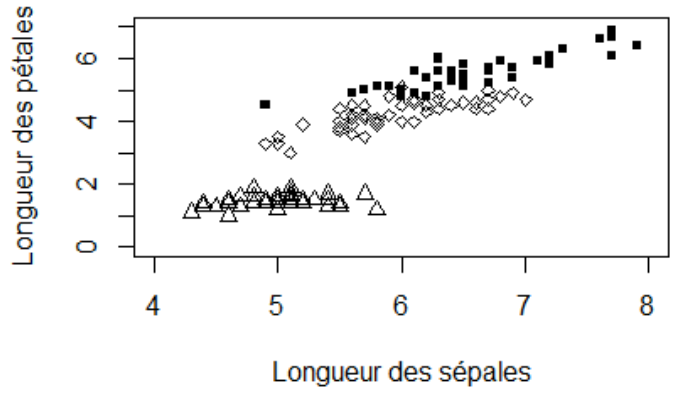
6.2.2. To add points or lines

- `points()`. *First a graphic for Iris setosa then the addition of the points that corresponds to other species with for the points of different forms*

```
>plot(Lpetale~Lsepale, irisdata, xlim=c(4,8),
ylim=c(0,7),type='n',xlab= ' Longueur des sépales ', ylab=
'Longueur des pétales')
>points(Lpetale~Lsepale,subset(irisdata,Especes=="setosa"),
pch=24)
```

```
>points(Lpetale~Lsepale,subset(irisdata,Especes=="virginica"),
cex=0.7, pch=15)
>points(Lpetale~Lsepale,subset(irisdata,Especes=="versicolor")
, cex=0.7, pch=5)
```

⚠ It is important to set the scale of x and y in the plot () when you call it for the first time. If you don't do it, you won't see the overlay as R will set the x and y by default from the first graph and do not readjust it according to the new superposition.

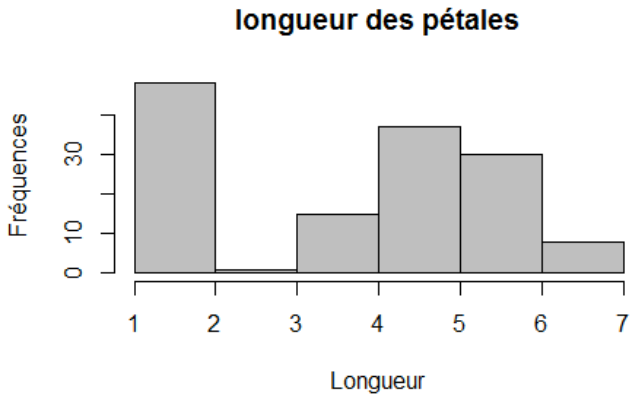


.

6.3. The function *hist()*

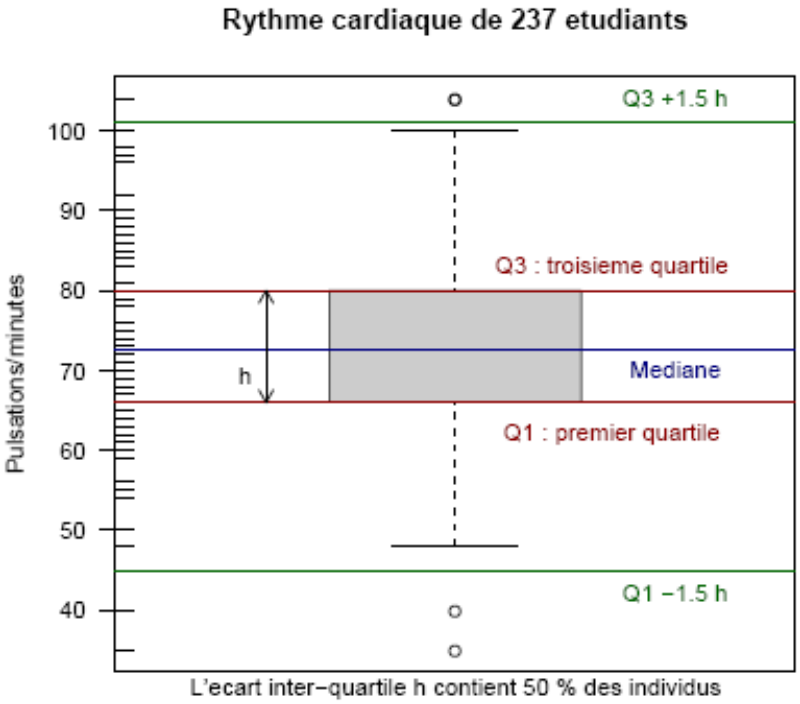
Histograms are useful to display the frequency of a variable. The argument `breaks` is to set the number of classes. The rest of the arguments are the same as plot().

```
> hist(irisdata$Lpetale, breaks
   = 4)
> hist(irisdata$Lpetale, breaks
   = c(1,2,3,4,5,6,7),
   main="longueur des pétales",
   col="grey",
   xlab= « Longueur »,
   ylab= « Fréquences »)
```

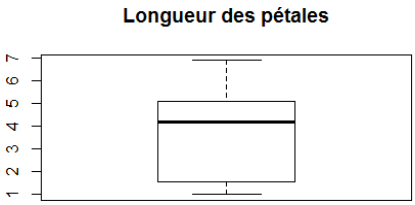


6.4. The function *boxplot()*

Extract for Wikipedia "*They are useful for descriptive statistics, a box plot or boxplot is a convenient way of graphically depicting groups of numerical data through their quartiles. Box plots may also have lines extending vertically from the boxes (whiskers) indicating variability outside the upper and lower quartiles, hence the terms box-and-whisker plot and box-and-whisker diagram. Outliers may be plotted as individual points. Box plots are non-parametric: they display variation in samples of a statistical population without making any assumptions of the underlying statistical distribution. The spacing between the different parts of the box indicate the degree of dispersion (spread) and skewness in the data, and show outliers. In addition to the points themselves, they allow one to visually estimate various L-estimators, notably the interquartile range, midhinge, range, mid-range, and trimean. Box plots can be drawn either horizontally or vertically*."

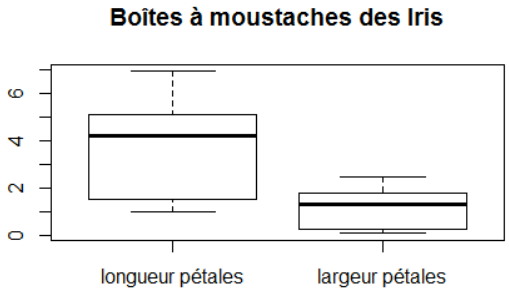

Rythme cardiaque de 237 etudiants
Q3 +1.5 h
Q3 : troisieme quartile
Mediane
Q1 : premier quartile
Q1 −1.5 h
h
Pulsations/minutes
100
90
80
70
60
50
40
L'ecart inter−quartile h contient 50 % des individus

```
>   boxplot(irisdata$Lpetale,
main="Longueur des pétales")
```



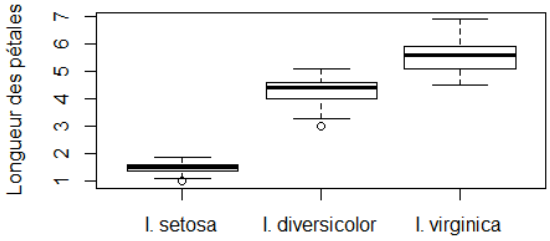
You can display several boxplots on the same graphic for several variables? The argument names can help to specify a name for each of them. For example, *Petal.Length, Sepal.Length, Petal.Width et Sepal.Width.*

```
>   boxplot(irisdata$Lpetale,
irisdata$larpetale,   main   =
"Boîtes   à   moustaches   des
Iris",      names=c("longueur
pétales", "largeur pétales"))
```



You can represent several boxplots according to the modality af a variable. For example, the 3 species *of Iris* to get the lengths of theirs sepals.

```
>   boxplot(irisdata$Lpetale   ~
irisdata$Especes,      ylab      =
"Longueur      de      pétales",
names=c("I.      setosa",      "I.
diversicolor", "I. virginica"))
```



6.5. Examples with other functions

• **The scatter plot or diagram** (axis of sorted quantitative data): function `stripchart()`

• **Pie:** function `pie()`

⚠ Have a look to

`http://www.duclert.org/Aide-memoire-R/Graphiques/Parametres-des-graphes.php`

7. Exercises

Exercise 1.

1.1. Create a vector of number
- the list from 1 to 100
- A number list/sequence for 10, 20, 25, 50, repeated 5 times *(10 20 25 50 10 20 25 50 10 20 25 50 10 20 25 50 10 20 25 50)*
- a list from 1 to 100, with a step of 5
- Repeat 10 times the number 12

1.2. A vector with the number 1, 2, 3 with a repetition of each 4 times with in addition a repetition of the sequence of 4 times
- Put this vector in the object « VEC »
- Multiply all the element of VEC by 5
- Calculate the median and the quantile of 75%

1.3. Create a vector with a serie of number from 1 to 2000 with a step of 10
- Put it in the object « vec »
- Extract the 10^{th} value of vec
- Display a sub-vector called « vec2 » that corresponds to the values from 2 to 6
- Replace the last value of vec2 by 100
- Display vec2 without its 3th value and store it in vec3
- Replace all the values >= 30 by 30

Exercise 2. Operations with tables

Create a matric with 5 columns and 20 lines with the values from 1 to 100, by lines
-Call it MAT
-Display the value at the second line and 5^{th} column et replace it by NA
-Create "mat " with only the columns 2,3 and 4 (taking them from MAT)
-Replace the values between 40 and 60 by 50 in the matrix « mat »

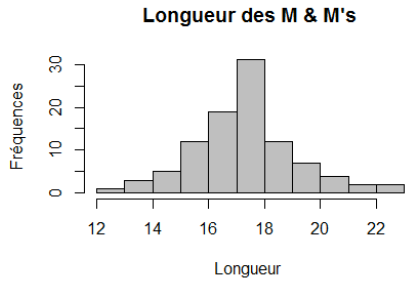
Exercise 3. Operations with a table of decimal numbers

3.1. Get the file (moodle) M&Ms.xls and open it with excel or open-office
- save it as MetMs.txt
- Import it in R and store it in data

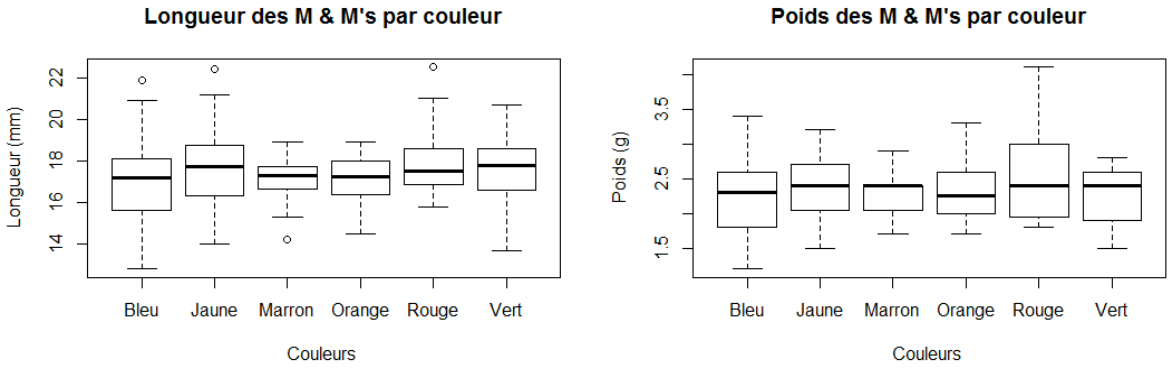
3.2. Create data2 by removing the variable largeur (width) from MetMs and the missing data
- Determine for which colors the lengths are minimal and maximal
- Sort the lines of data2 according to an increasing order of the length

Exercise 4. Graphics

4.1. Recopy this graphic with `data2`



4.2. Recopy the following graphics side by side



ANNEX 1 : basic commands

Commands	Description
`install.packages («namepackage»)`	To install the package namepackage
`library(nompackage)`	To load the package
`help.search(«testt»)`	To display the help
`?fonction`	To get information for the function
`help.start()`	Help online
`read.table()`	To import a file.txt in R
`write.table()`	To export data in a file.txt from R

FICHE 2 : OPERATIONS USUELLES / VECTEURS / MATRICES

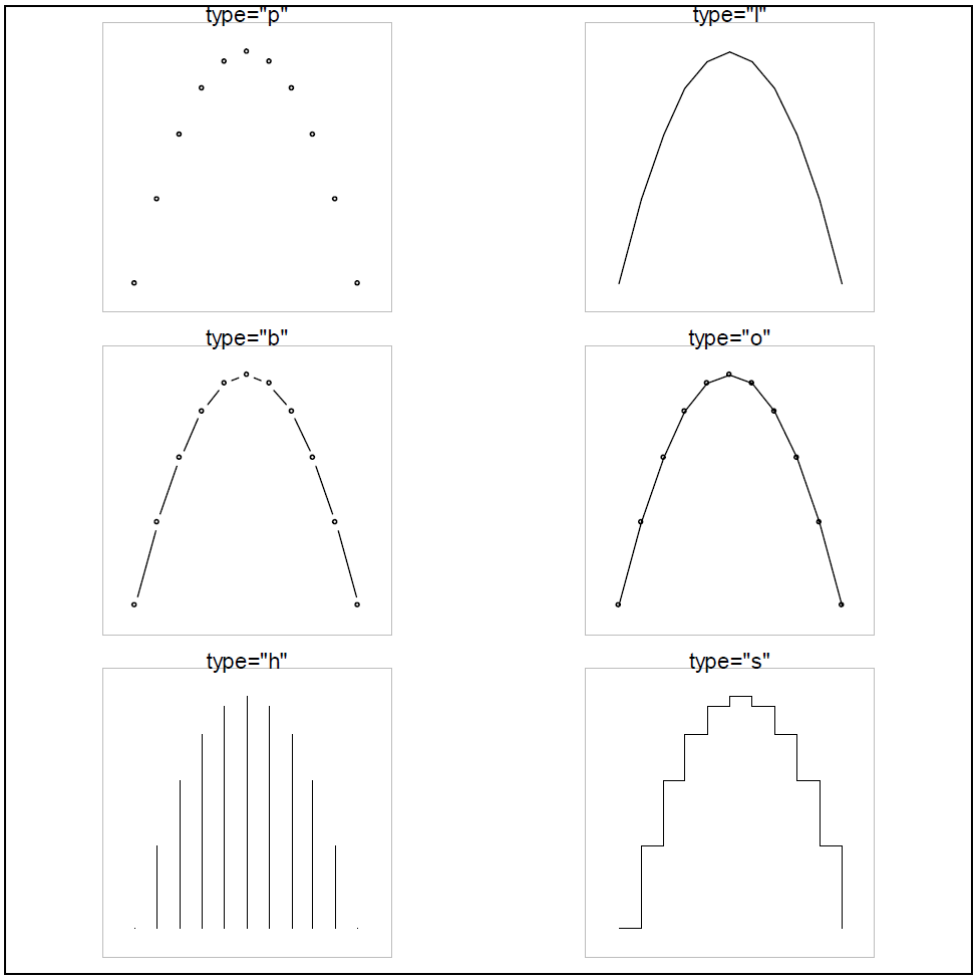
Opérations usuelles	Description
`+ , -`	Addition, subtraction
`* , /`	Multiplication, division
`exp()`	Exponential of a number
`log10(), log2()`	Logarithm of base 10, 2
`mean(vec1)`	Mean of the elements of vec1
`median(vec1)`	Median of the elements of vec1
`quantile(vec1,0.25)`	Quartile of 25% of the elements of vec1
`sqrt()`	The square root of the elements of a vector
`sum()`	The sum of the elements of a vector
`sd()`	The standard deviation of the elements of a vector
`c(i,j,k)`	To create a vector with a dimension of 3, with i,j,k
`vec<- c(i,j,k)`	To give a new name
`vec[i]`	Display the ith value of a vector
`vec*j`	Multiplication of a vector by j
`mean(vec)`	Mean with all the values of a vector
`a:b`	To construct a vector with numbers between a and b
`seq(from = i, to = j, by = k)`	To construct a vector with numbers between i and j with a step of k
`rep(c(i,j,k), each = l, times = m)`	To construct a repeated sequence of c(i,j,k), each element is repeated once and the sequence m times
`mat<-matrix(vec,nrow=n, ncol=p,byrow=T)`	To construct a matrix mat from a vector vec, of n lines and p columns and by filling it by lines
`t(mat)`	Transposition of mat
`mat[i,j]`	Selection of en element
`mat[,j]`	Selection of the jith column
`mat[vec1<i & vec2>=j,]`	Selection of elements according to conditions
`mat[-c(k,l),]`	Suppression of lines
`mat[i,j]<-k`	Replacement of elements
`mat[mat<=i]<-k`	Replacement of elements according to conditions
`mat[order(vec1),]`	To sort, increasing order
`mat[rev(order(vec1)),]`	To sort, decreasing order
`summary(mat)`	Statistics
`names(mat)`	To display names
`attach(mat)`	To attach to directly access to variables
`detach(mat)`	To detach the object mat
`rbind (mat1, mat2)`	Vertical concatenation of mat1 and mat2 (have the same names of variables)
`cbind (mat1, mat2)`	Horizontal concatenation of mat1 and mat2 (have the same names of lines)
`scale(mat)`	Normalization of all the columns
`tapply(X, INDEX,FUN)`	To apply a function (FUN) by category (INDEX) on X

ANNEX 3 : The graphics

Graphical Functions	Description
`plot(y~x)`	Simple graphic of y according to x Argument `axes=FALSE` -> suppress the rectangle Created by the axis
`hist(x, breaks=k)`	Histogram of frequency for X `breaks` = k to set the number of classes
`boxplot(x,y,z)`	Boxplots of x,y,z
`boxplot(x~k)`	Boxplot of x according to the modality of k
`stripchart(x)`	Scatter diagram
`pie(x)`	Pie
`+axis(nombre)`	Addition of an axis
`+title()`	Addition of titles
`+abline()`	Addition of a curve
`+points(x,y)`	Addition of supplementary points
`+lines(x,y)`	Addition of supplementary lines

Arguments	Description
`main= « nom_graphe »`	To add a main title
`xlab =« titre_axex »`	To add a legend
`ylab=« titre_axey »`	To add a legend
`xlim=c(nombre_min, nombre_max)`	To set a scale
`ylim=c(nombre_min, nombre_max)`	To set a scale
`lty = nombre_type_ligne`	To choose the type of the line: *1=continue line; 2=dotted line*
`lwd = nombre_epaisseur_ligne lwd.thicks`	To choose the thickness
`col = « couleur »`	To choose the color of a line or symbol
`pch= nombre_type_symbol`	To choose the symbols
`cex= nombre_taille_symbol`	To choose the size of the symbols
`type= « type_ligne »`	To choose the type of lines

Type of lines with the argument `type`



Type of symbols with the argument `pch`