Introduction to R

Fundamentals in statistics

C. Dillmann

august 2022

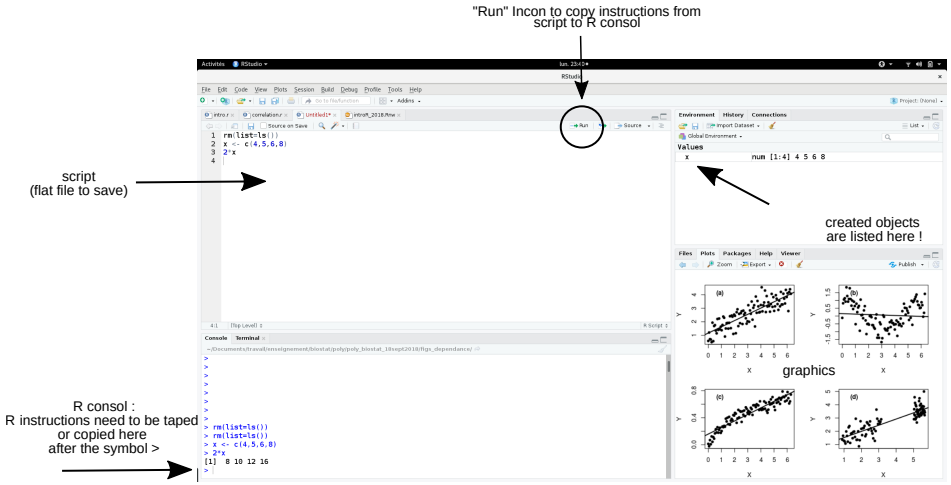
What is R ?

The R software is :

- An environment for statistical and numerical analyses
- A tool to realize graphical outputs
- A complete toolkit
- A programming language

The official website : http://www.r-project.org.

Interactions with R : The Rstudio software



Four panels to follow the work :

- **Console :** The command is to be written after > and then press *Enter*.
- **Script :** Open a new script : *File - New script*. Then write down the succession of instructions in the script and copy them in the console. Save the script !
- **Environment :** All the objects created during the session are listed.
- **Plots :** Fast vizualisation of graphical outputs.

Learning by practise

- Open Rstudio
- Choose a working directory containing the file *cinetik.txt*
- Reproduce the instructions from the following slides and try to understand them.

R, a calculator

Some examples :

```
> (1+3)*3-1
[1] 11
> exp(10)
[1] 22026.47
```

- Arithmetic operations : `+ - * / ^`.
- usual mathematical functions : `exp()`, `log()`, `cos()`...
- More information : `help('Math', package='base')`.

Variables and Affectations

As most programmation languages, R allows for symbolic manipulation through variables. To do this, use the affectation oprators : `<-` or `->` or `=`.

```
> x<-2
> 2*x+3->y
> s="This is a character string"
> x
[1] 2
> y
[1] 7
> s
[1] "This is a character string"
```

Variables names

Variables names are flexible and do not need to be declared (*i.e.*, a variable can store anly value: *integer*, *double*, *character*, ...). But you need to respect a few rules :

- Variable names cannot begin by a number nor a special character.
- Variable names are case sensitive (UPPER CASE/lower case).
- R uses reserved letters/words (*e.g.* `c`, `q`, `t`, `D`, ...) that need to be avoided as variable names Par exemple :

```
> NA=2
Erreur dans NA = 2
   membre gauche de l'assignation (do_set) incorrect
```

Variables names

Variables names are flexible and do not need to be declared (*i.e.*, a variable can store anly value: *integer*, *double*, *character*, ...). But you need to respect a few rules :

- Variable names cannot begin by a number nor a special character.
- Variable names are case sensitive (UPPER CASE/lower case).
- R uses reserved letters/words (*e.g.* `c`, `q`, `t`, `D`, ...) that need to be avoided as variable names Par exemple :

```
> NA=2
Erreur dans NA = 2
   membre gauche de l'assignation (do_set) incorrect
```

Variables names

Variables names are flexible and do not need to be declared (*i.e.*, a variable can store anly value: *integer*, *double*, *character*, ...). But you need to respect a few rules :

- Variable names cannot begin by a number nor a special character.
- Variable names are case sensitive (UPPER CASE/lower case).
- R uses reserved letters/words (*e.g.* `c`, `q`, `t`, `D`, ...) that need to be avoided as variable names Par exemple :

```
> NA=2
Erreur dans NA = 2
   membre gauche de l'assignation (do_set) incorrect
```

Variables names

Variables names are flexible and do not need to be declared (*i.e.*, a variable can store anly value: *integer*, *double*, *character*, ...). But you need to respect a few rules :

- Variable names cannot begin by a number nor a special character.
- Variable names are case sensitive (UPPER CASE/lower case).
- R uses reserved letters/words (*e.g.* `c`, `q`, `t`, `D`, ...) that need to be avoided as variable names Par exemple :

```
> NA=2
Erreur dans NA = 2
   membre gauche de l'assignation (do_set) incorrect
```

Particular values

- `NULL` : null or empty object.
- `NA` : *Not Available*.
- `NaN` : *Not a Numeric*.
- `-inf`/`inf` : Infinity.
- `TRUE`/`T` : True
- `FALSE`/`F` : False.

Objects

R works on data structures called objets. Objects can be manipulated via algebric rules, logical instructions, or functions.
An object is characterized by

- its name : a simple or composed word without spaces.
- its mode : describes the nature of the object stored : numerical values, characters, booleans, factors.
- One or several values : NA stands for missing data.
- its type : scalar, vector, table (data.frame) or list.

Objets

R works on data structures called objets. Objects can be manipulated via algebric rules, logical instructions, or functions.
Examples :

Exemple

```
> my.vector=c(1,12,18.4,7,NA)
> is.vector(my.vector)
[1] TRUE
> is.character(my.vector)
[1] FALSE
> is.na(my.vector)
[1] FALSE FALSE FALSE FALSE  TRUE
```

Scalars and vectors

a scalar is a single value (integer or float):

```
> x=2.3
[1] 2.3
```

In R, elementary types are vectors, that can be created using different ways:

```
> c(12, 23, 98, 12.2, 20)
[1] 12.0 23.0 98.0 12.2 20.0
> rep(1, 4)
[1] 1 1 1 1
> seq(1,2, by=0.1)
[1] 1.0 1.1 1.2 1.3 1.4 1.5 1.6 1.7 1.8 1.9 2.0
> 3:9
[1] 3 4 5 6 7 8 9
```

Arithmetics on vectors

- Usual operators `+ - * / ^` (term by terms):

```
> 1:5+5:1
[1] 6 6 6 6 6
```

- Mathematical functions (term by term) :

```
> log10(c(10,100,1000))
[1] 1 2 3
```

- Special functions:

```
> length(1:10)
[1] 10
> sum(1:10)
[1] 55
```

Arithmetics on vectors

- Usual operators `+ - * / ^` (term by terms):

```
> 1:5+5:1
[1] 6 6 6 6 6
```

- Mathematical functions (term by term) :

```
> log10(c(10,100,1000))
[1] 1 2 3
```

- Special functions:

```
> length(1:10)
[1] 10
> sum(1:10)
[1] 55
```

Arithmetics on vectors

- Usual operators `+ - * / ^` (term by terms):

```
> 1:5+5:1
[1] 6 6 6 6 6
```

- Mathematical functions (term by term) :

```
> log10(c(10,100,1000))
[1] 1 2 3
```

- Special functions:

```
> length(1:10)
[1] 10
> sum(1:10)
[1] 55
```

Booleans and logical operators

- Special values: `TRUE`/`FALSE`.
- are the result of a boolean operation :

```
> 1>2
[1] FALSE
```

- Comparison operators: `==`, `!=`, `<`, `>=`, ...
- Combine instructions: `&` (et), `|` (ou) :

```
> x=c(10, 23, 3, 30)
> x<9 | x>25
[1] FALSE FALSE  TRUE  TRUE
```

Extract data from a vector

- From the indices:

```
> x=c(1,3,5,7,9,12,14,18,22)
```

  1. Choose the elements to be kept

```
> x[1:3]
[1] 1 3 5
```

  2. Or choose the elements to remove

```
> x[-c(1,2,3)]
[1] 7 9 12 14 18 22
```

- From a boolean vector :

```
> x[x>=9]
[1]  9 12 14 18 22
```

Extract data from a vector

- From the indices:

```
> x=c(1,3,5,7,9,12,14,18,22)
```

  1. Choose the elements to be kept

```
> x[1:3]
[1] 1 3 5
```

  2. Or choose the elements to remove

```
> x[-c(1,2,3)]
[1] 7 9 12 14 18 22
```

- From a boolean vector :

```
> x[x>=9]
[1]  9 12 14 18 22
```

Categorical variables: the factor mode

With R, a qualitative variable is interpreted as a factor. R associates to such a variable a vector that contains the possible values(`levels`). The mode of anly variable can be changed into a factor though the function `as.factor()`

```
> group=c("A","A","B","C")
> group
[1] "A" "A" "B" "C"
> group=as.factor(group)
> group
[1] A A B C
Levels: A B C
> levels(group)
[1] "A" "B" "C"
```

Data tables or data.frame

Are particular objects where each column can have a different mode.

- Reading a data frame:
  ```
  > data=read.table("cinetik.txt", header=TRUE)
  ```
- Extracting a column from the data frame:
  ```
  > colnames(data)
  [1] "manip"  "souche" "temps"  "temoin" "eff"
  ```
  To see the column *manip*, type : `data$manip`
- Extract some elements of a column: see extract elements from a vector.
- Special data.frame functions: `summary()`, `table()`, `head()`, `dim()`, ...

The cinetik.txt data.frame

The data contain the results from the growth of different *E. Coli* strains with or without an effector. There were five different batches (*manip*). *E.Coli* population size was estimated by the optical density of the culture. For each strain in each batch, the OD was taken at the same time (*temps*) for the culture without (*temoin*) or with (*eff*) the effector. Data were provided by *francoise.budar@inrae.fr*

Load the data frame and use the instructions of this tutorial to represent the growth of the five stains in the two different conditions.

Lists

Lists are collections of objects that can differ both for thair mode and their dimension.
For example, one may wish to strore in the same object the following vectors:

```
> students=c("Céline","Valérie","Thomas","Julien")
> math=c(12,14,18,13)
> french=c(11,15,10)
> presents=c(T,T,F,F,F,T)
> my.list=list(students,math,french,presents)
```

Lists

... that gives,

```
> my.list
[[1]]
[1] "Céline"  "Valérie" "Thomas"  "Julien"

[[2]]
[1] 12 14 18 13

[[3]]
[1] 11 15 10

[[4]]
[1]  TRUE  TRUE FALSE FALSE FALSE  TRUE
> my.list[[2]]
[1] 12 14 18 13
```

Lists

The elements from a list can be named:

```
> names(my.list)=c("students","math","french","presents")
> my.list
$students
[1] "Céline"  "Valérie" "Thomas"  "Julien"

$math
[1] 12 14 18 13

$french
[1] 11 15 10

$presents
[1]  TRUE  TRUE FALSE FALSE FALSE  TRUE
```

Lists

To see the vector *math* from the list *my.list*, type : `my.list$math`

```
> my.list$math
[1] 12 14 18 13
```

Functions

A function is defined by:

- Its name (the instruction `ls("package:base")` shows the list of R basic functions).
- Series of arguments (Type `args(nom_fonction)` to see the arguments of a function).
- The returned object, which is generally a list.
- A help section (Type `help(chisq.test)` for example)

Create your own function :

```
my_function_name=function(arg1, arg2, ...){
  result=sum(arg1)+prod(arg2) Series of instructions
  result Send the result
}
```

Functions

When a function is called without being given a name, R shows a synthetic report.

Example :

```
> chisq.test(c(23,52,22))

        Chi-squared test for given probabilities

data:  c(23, 52, 22)
X-squared = 17.959, df = 2, p-value = 0.000126
```

Functions

When the result of a function is attributed to a named object, the object contains as a list all the values calculated by the function.

Example :

```
> result=chisq.test(c(23,52,22))
> attributes(result)

$names
[1] "statistic" "parameter" "p.value"   "method"    "data.name
[7] "expected"  "residuals" "stdres"


$class
[1] "htest"

> result$p.value

[1] 0.0001259807
```

The arguments of a function

The list of a function arguments are stored in the Usage section from the function help.

Exctract of the help section for the t.test function

```
t.test(x, y = NULL,
       alternative = c("two.sided", "less", "greater"),
       mu = 0, paired = FALSE, var.equal = FALSE,
       conf.level = 0.95, ...)
```

- The first argument is always the name of the object to which the function is applied.
- Check whether it should be a vector, a list or a data.frame.
- Following arguments are the options, with default values.

Function's arguments

An explanation is given in the Arguments subsection of the help section.

Function t.test

```
alternative: a character string specifying the alternative
          hypothesis, must be one of '"two.sided"' (default),
           '"greater"' or'"less"'.
           You can specify just the initial letter.
```

- It is not necessary to specify all the arguments when calling a function.
- Non specified arguments will take their default value.
- When calling a function, the arguments need either to be given in the same order as in the help section, or need to be named.

Calling a function

```
t.test(x, y = NULL,
       alternative = c("two.sided", "less", "greater"),
       mu = 0, paired = FALSE, var.equal = FALSE,
       conf.level = 0.95, ...)
```

```
> x=rnorm(20,mean=5)
> y=rnorm(30,mean=10)
> t.test(alternative="less",var.equal=T,x=x,y=y)

        Two Sample t-test

data:  x and y
t = -17.182, df = 48, p-value < 2.2e-16
alternative hypothesis: true difference in means is less than
95 percent confidence interval:
      -Inf -4.472772
sample estimates:
mean of x mean of y
```

Function's results

The list of objects returned by a function are in the Value subsection of the help section.

t.test fuction

```
Value:
     A list with class '"htest"' containing the
     following components:
 statistic: the value of the t-statistic.
 parameter: the degrees of freedom for the t-statistic. ...
```

- To get all results from a function, you need to create a named oject.
  `>my.ttest=t.test(x,y,paired=T)`
- `my.ttest` is a list which elements can be seeked for. For example `my.ttest$parameter` gives the value of the t.test statistics.

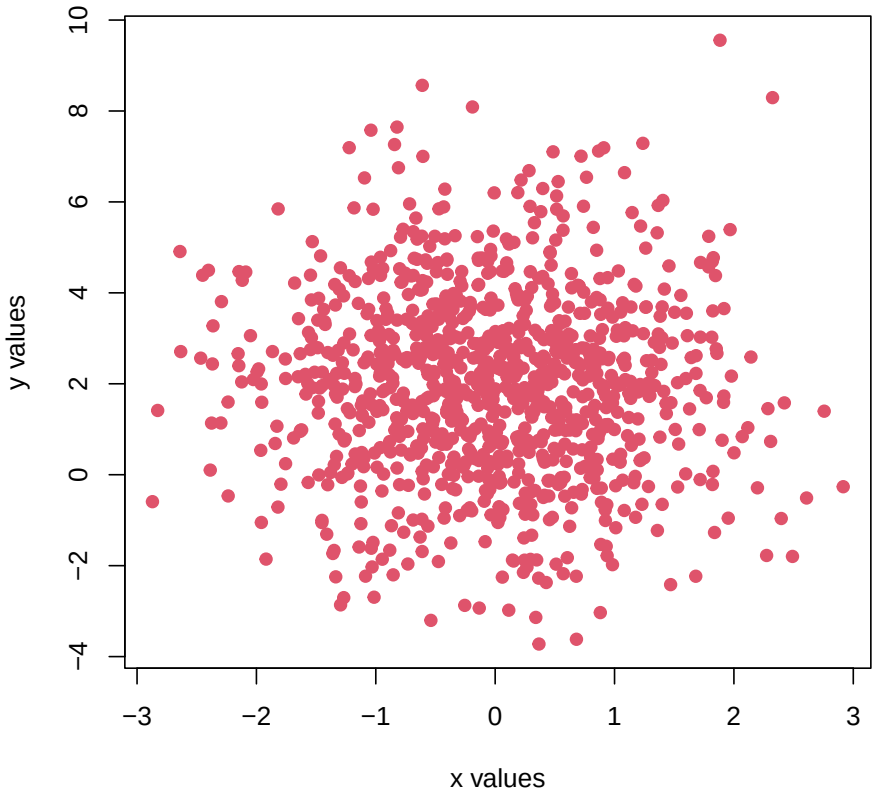
Plots

First, let's create three random vectors,

```
> x=rnorm(1000)
> y=rnorm(1000,m=2,sd=2)
> z=rnorm(1000,m=3,sd=4)
```

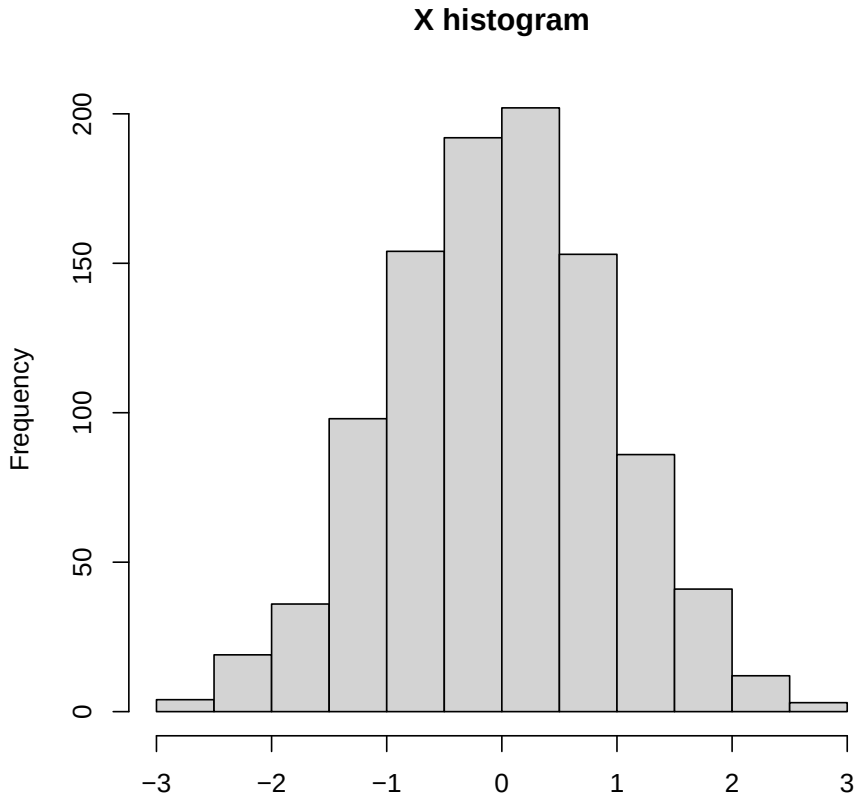
Scatterplots : *plot()*

```
plot(x,y,xlab="X values",ylab="Y values"
                ,pch=19,col=2)
```



Histograms : *hist()*

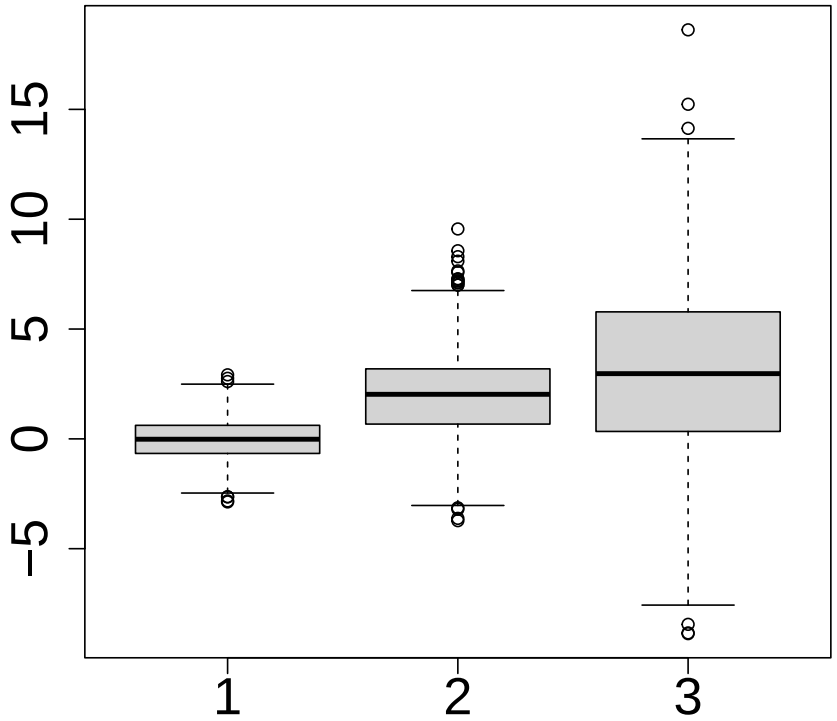
```
hist(x,20,main="histogramme de X",xlab="")
```



Here, the second argument defines the number of classes.

Boxplots: *boxplot()*

```
boxplot(list(x,y,z),cex.axis=2)
```



Here, the axes labels are twofold larger.

Conclusion

- R langage has become a standard for data analysis.
- As all langages, you need to learn the vocabulary and the grammar.
- R can also be used to handle databases and numerical equation solving.
- It is a free software that comes with its pros/cons.

Have fun !