LABORATOIRE HUBERT CURIEN
UMR • CNRS • 5516 • SAINT-ETIENNE
UNIVERSITÉ DE LYON

From Statistics to Data Mining

Master 1
COlour in Science and Industry (COSI)
Cyber-Physical Social System (CPS2)
Saint-Étienne, France

Fabrice MUHLENBACH
https://perso.univ-st-etienne.fr/muhlfabr/
e-mail: fabrice.muhlenbach@univ-st-etienne.fr

Introduction to R

- Use of R

Why R ?

- evolution :
  statistical software for specialists (statisticians)
  → free software environment for statistical computing used by everyone
- success due to a strong community of developers
- allows you to go beyond the limits of spreadsheets for data analysis (number of rows and columns, features, etc.)
- allows data analysis without having to program dozens of lines of code in a classic computer language
  (→ saving time)

• What is  ?

- a programming language for data analysis
- history: language ***S*** (Bell labs) in 1976
- evolution: ***S-plus*** (1988), software for specialists
- creation of ***R***, open-source clone version of ***S*** in 1996
  R-Core Team (University of Auckland), small community
- base + packages (around 17,000) distributed by the CRAN (*Comprehensive R Archive Network*), https://cran.r-project.org/web/packages/
- ***R*** can be run via the command line console (console), but it can also be run using an IDE like ***RStudio***

• What is R ? (continued)

- ***R*** is an interpreted language (possible compilation)
- ***R*** paradigm: array, object-oriented, imperative, functional, procedural, reflective
- non-typed language
- vector and table size can be resizable
- interactive use
- everything in a ***R*** is an object, and the operations made on these objects are vectorized:

```
> 3 + 1
[1] 4

> 1:3 + 1
[1] 2 3 4
```

• What is  ? (continued)

- why work on vectors? experimental data is coming in abundance, so it is normal to treat them as vectors
- less programming with ***R*** because we reuse functions already programmed (packages)
- **R** works well on Windows, macOS or Linux operating systems
- ***R*** is free and distributed under the GPL (GNU General Public License)
- ***R*** is easily installed using an executable installation file from the CRAN site or local mirror sites: https://cran.r-project.org/mirrors.html

• Advantages of using R (*vs*. a spreadsheet)

Limitations of a spreadsheet:

- in a spreadsheet, the number of rows is limited
- mix of data and calculations: difficult to understand
- the recalculation of a sheet is not linear
- not optimal calculation and data storage
- unstructured data
- from the statistical point of view, the graphs and the calculation functions present in a spreadsheet are weak (in choice, in options, in comprehension ...)
- automating some tasks
  → need to use macro-commands

Working with variables:

- assignment of a value to a variable:
  ```
  x <- 1
  x = 1
  ```
- numeric variable: `x = 10.5`
- display of the variable category:
  ```
  class(x)
  [1] "numeric"
  ```
- the variables can alse be logical (`TRUE` / `FALSE`), complex (`1 + 2i`), strings of characters, vectors, matrices, lists, *data frames* or other kind of objects (graphs, images, connections to a database…)

Working with variables:

- arithmetic operators:
  addition: `+`
  subtraction: `-`
  multiplication: `*`
  division: `/`
  exponentiation: `^` or `**`
  modulo: `%%`
  integer division: `%/%`
- logical operators:
  smaller than: `<`
  greater than: `>`
  smaller than or equal to: `<=`
  greater than or equal to: `>=`
  equality : `==`
  different : `!=`
  no: `!`
  and: `&`
  or: `|`

Programming instructions:

- tests :

```
if (condition){
    instructions if the condition is true
    } else {
    instructions if the condition is false
    }
```

- example:

```
x <- 1
if (x == 1){
  "x equals 1"
  } else {
  "x is different of 1"
  }
```

Programming instructions:

- loop with a known number of iterations:
  `for (`variable `in` list`)` instruction
- example 1 :

```
for (i in seq(from=5, to=15, by=5))
print(i)
[1] 5
[1] 10
[1] 15
```

- example 2 :

```
for (i in 1:3) print(i * 4)
[1] 4
[1] 8
[1] 12
```

Programming instructions:

- loop with an unknown number of iterations :
  `while (condition) instruction`
- example :

```
i <- 0
while (i <= 25) {
  print(i)
  i <- i + 10
  }
```

- results:

```
[1] 0
[1] 10
[1] 20
```

Programming instructions:

- define and use a function:
  `function(`list of parameters`)` expression
  `return(`value`)`
- example of creating a function:

```
maFonction <- function(a,b,c,x){
   y <- a * x^2 + b * x + c
   return(y)
}
```

- using this function:

```
> maFonction(1, 2, 3, 4)
[1] 27
> maFonction(4, 3, 2, 1)
[1] 9
```

Storing data in a data frame:

- *data frame* = specific data structure
- format: *x* rows × *y* columns corresponding to:
  x individuals × *y* variables
- list of vectors of the same length but of (possibly) different formats: integer, real, logic, factor...
- most used format for data processing
- the variables have names, which allows you to access specific data by indicating a specific index
- use:
  - numeric index to indicate a row or a column
  - name of the variable preceded by the symbol $

Storing data in a data frame:

- creation of a data frame

```
v1 <- c(14, 8.5, 13.8, 15)
v2 <- (v1 >= 10)
v3 <- factor(c("M", "F", "M", "F"))

# creation of a list called "liste"
liste <- list(v1,v2,v3)

# creation of a data frame called "donnees"
donnees <- data.frame(liste)

# naming the columns and the rows
colnames(donnees) <- c("Note","Reussite","Genre")
rownames(donnees) <- c("Pierre","Pauline","Jacques","Mathilde")
```

Storing data in a data frame:

- display the data

```
> print(class(donnees))
[1] "data.frame"
> print(summary(donnees))
      Note          Reussite        Genre
 Min.   : 8.50   Mode :logical   F:2
 1st Qu.:12.47   FALSE:1         M:2
 Median :13.90   TRUE :3
 Mean   :12.82   NA's :0
 3rd Qu.:14.25
 Max.   :15.00
> print(donnees)
         Note Reussite Genre
Pierre   14.0     TRUE     M
Pauline   8.5    FALSE     F
Jacques  13.8     TRUE     M
Mathilde 15.0     TRUE     F
> donnees$Genre[2]
[1] F
Levels: F M
```

Storing data in a data frame:

- selection of the data

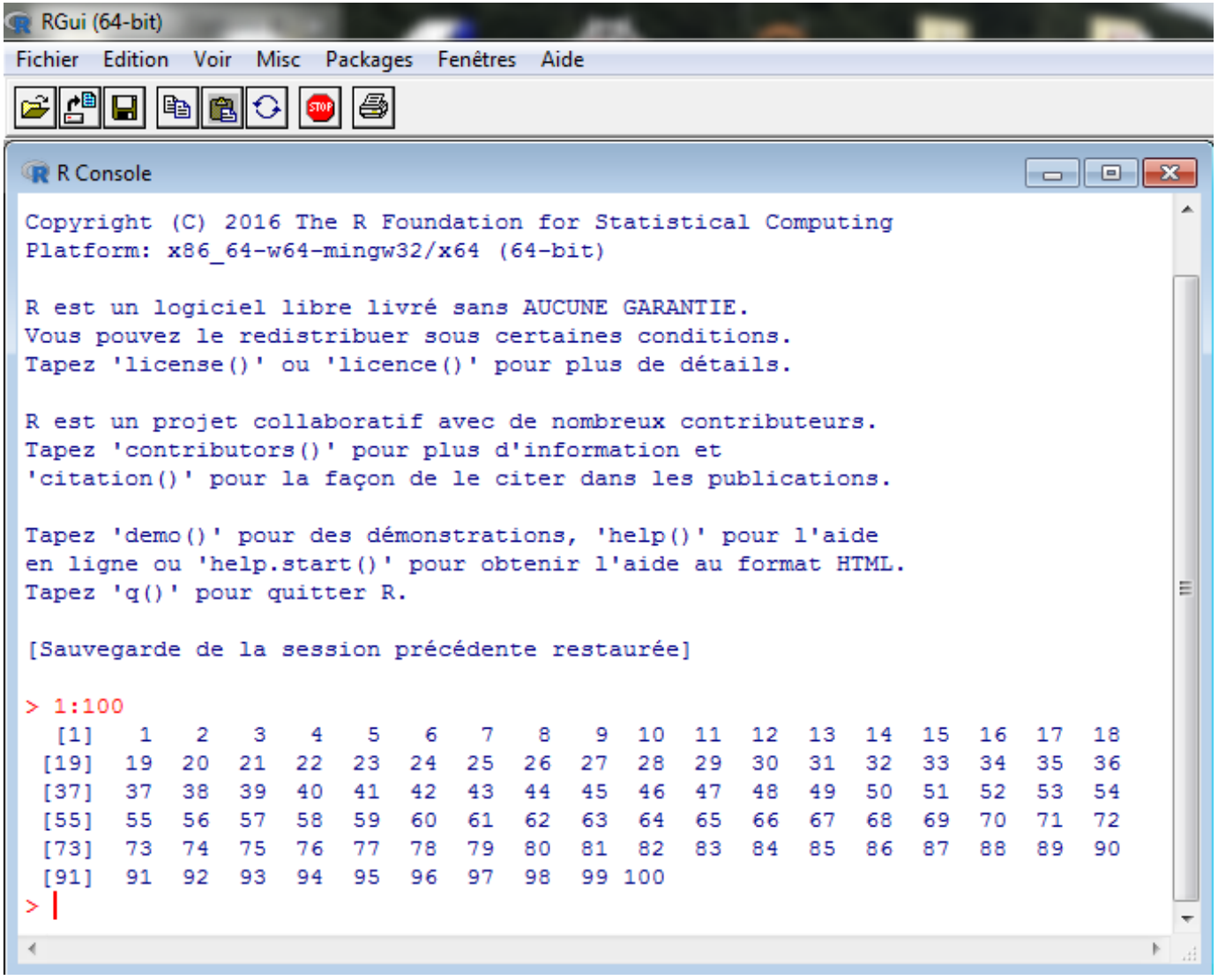
```
> # Sélection d'un individu (ligne) par indice
> donnees[1,]
       Note Reussite Genre
Pierre   14     TRUE     M
> # Sélection d'une variable (colonne) par indice
> donnees[,1]
[1] 14.0  8.5 13.8 15.0
> # Sélection d'une variable par nom de variable
> donnees$Note
[1] 14.0  8.5 13.8 15.0
```

Storing data in a data frame:

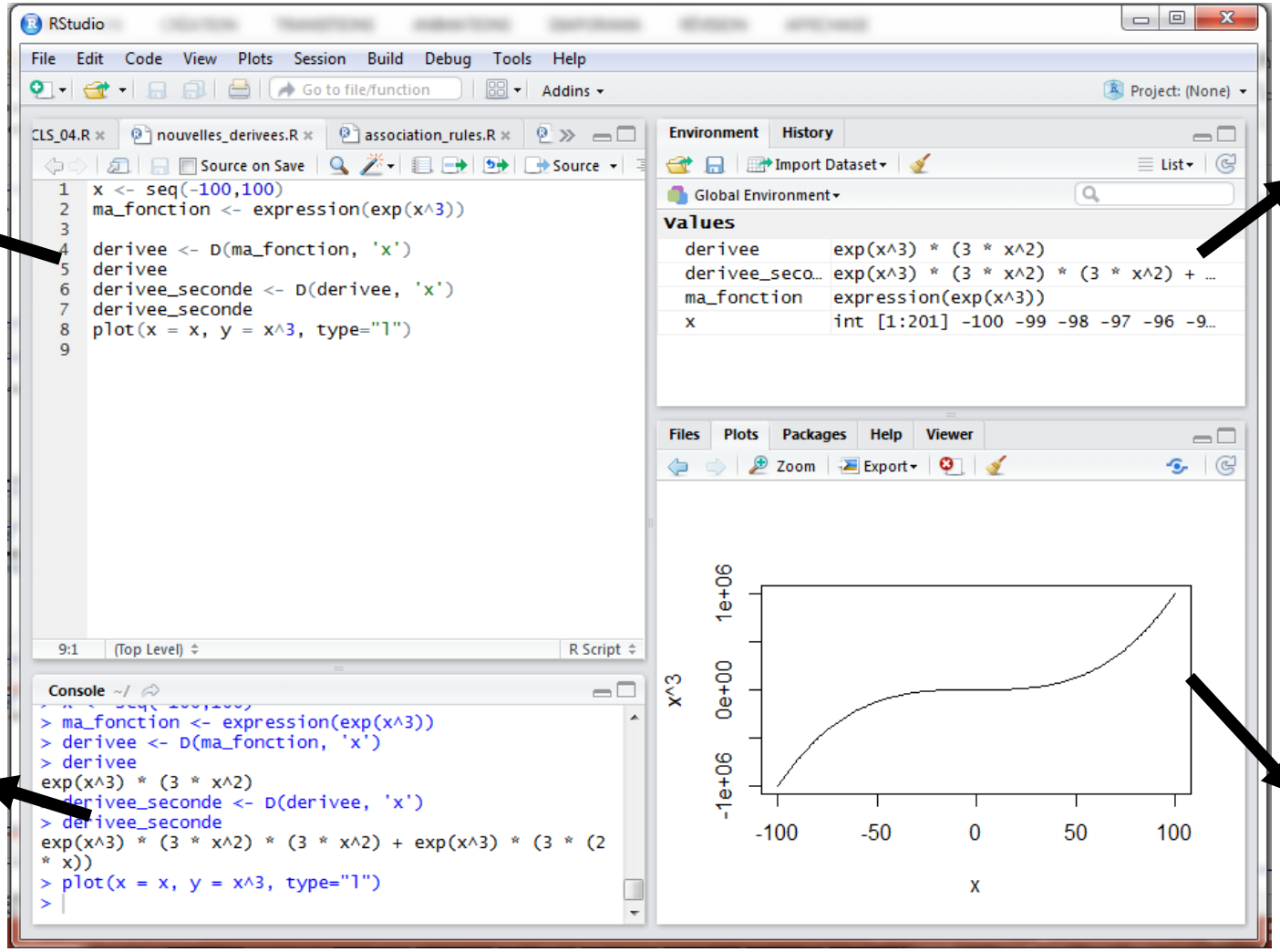
- selection of the data with conditions

```
> # Display the people whose gender is male:
> donnees[donnees$Genre == "M",]
        Note Reussite Genre
Pierre  14.0     TRUE     M
Jacques 13.8     TRUE     M
> # Display the people with true success value:
> donnees[donnees$Reussite == "TRUE",]
         Note Reussite Genre
Pierre   14.0     TRUE     M
Jacques  13.8     TRUE     M
Mathilde 15.0     TRUE     F
> # or, more simply:
> donnees[donnees$Reussite,]
         Note Reussite Genre
Pierre   14.0     TRUE     M
Jacques  13.8     TRUE     M
Mathilde 15.0     TRUE     F
```

• R programming environment -**Console mode**



• programming environment –RStudio IDE



script editor

console

file variables functions

graphs packages help