Statistics for Biology
MSc. Computational Biology and Bioinformatics

Regina Bispo

Department of Mathematics, NOVA SST

2021/22

cma FCT.UNL
centro de matemática
e aplicações

FACULDADE DE
CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA
DEPARTAMENTO DE
MATEMÁTICA

NOVA
NOVA SCHOOL OF
SCIENCE & TECHNOLOGY

Week 1

Summary: Presentation.

Practical: Introduction to R and RStudio.

Presentation

Summary and Evaluation

- **Description**

  1. In this curricular unit we intend to explore some of the main methods of statistical modeling covering theory, applications and software.
  2. These methods will allow the student to deal with different type of variables (for example, continuous, counts, and presence/absence) and deal with observations indexed in time and/or space, and thus go beyond the standard statistical modeling techniques.
  3. Frequently used multivariate statistical methods are also introduced.

- **Evaluation**:

  1. Continuous evaluation: It includes the following components
     1. 1st mini-test (MT1)- Test to be carried out in a computer lab, in person, with a weight of 25%. The test will last for 2 hours. The test is rated on a scale of 0 to 20 points (no minimum rating).
     2. 2nd mini-test (MT2)- Test to be carried out in a computer lab, in person, with a weight of 25%. The test will last for 2 hours. The test is rated on a scale of 0 to 20 points (no minimum rating).
     3. Project (T)- Individual work of data analysis. The work will have a weight of 50%. The work is rated on a scale of 0 to 20 values.

     Final grade calculation formula (NF): $\text{NF} = 0.25 \times (\text{MT1} + \text{MT2}) + 0.5 \times \text{T}$ (partial grades rounded to 1 decimal place)
  2. Resource (Improvement)/Special evaluation: In-person written test to be held on a single date, within the period provided for in the academic calendar, with a weight of 100%. The exam will last for 3 hours. The exam is rated on a scale from 0 to 20 points.

Syllabus

Week 1	Presentation. Introduction to R and RStudio
Week 2	1. Basic concepts of Statistics 1.1 Sampling and experimental design techniques 1.2 Exploratory data analysis
Week 3	1.3 Statistical inference. Parameter estimation and hypothesis testing
Week 4	2. Linear and Generalized Linear Models 2.1 Continuous variables: linear regression
Week 5	2.2 Continuous variables: ANOVA/ANCOVA
Week 6	2.2 Continuous variables: Non-normal response variables
Week 7	2.3 Discrete variables: logistic regression
Week 8	2.3 Discrete variables: Poisson regression
Week 9	2.4 Generalized Linear Mixed Models (GLMM) Fixed vs. random effects
Week 10	Revisions and First evaluation
Week 11	3. Introduction to Multivariate Statistics 3.1 MANOVA
Week 12	3.2 Principal Component Analysis (PCA)
Week 13	3.3 Cluster analysis
Week 14	Revisions and Second evaluation

Bibliografia

- Crawley, M. (2012). The R Book. John Wiley & Sons.
- **Draper, N. R. & Smith, H. (1998) Regression Analysis, 3rd Edition Wiley**
- **Faraway, J. J. (2006) Extending the Linear Model with R. Chapman & Hall / CRC**
- Faraway, J. J. (2002) Practical Regression and Anova using R. e-book
- **Johnson, R. and Wichern, D. W. (2007), Applied Multivariate Statistical Analysis, 6th Edition, Prentice Hall, New Jersey**
- Montgomery, D. (2009). Design and Analysis of Experiments. John Wiley & Sons.
- Murteira, B., Ribeiro, C.S., Silva, J.A. & Pimenta C.(2002). NA NA
- Scheiner, S. M. & Jessica, G.(Eds) (2001). Design and Analysis of Ecological Experiments. Oxford University Press. NA
- Wood, S. N. (2006) Generalized Additive Models: an introduction with R. CRC/Chapman & Hall.
- Zuur, A. F.; Ieno, E. N.; Walker, N.; Saveliev, A. A. & Smith, G. M. (2009) Mixed effects models and extensions in ecology with R. Springer.
- Zuur, A.F., Elena N. Ieno & Erik H.W.G. Meesters (2009) A Beginner's Guide to R. Springer.
- Zuur, Ieno & Smith (2007). Analyzing Ecological Data. Springer.

Practical 1

Introduction to R and RStudio

What is R?

- R is a language and environment for statistical computing and graphics. It is an open-source created in 1995 **R**oss Ihaka and **R**obert Gentleman, Department of Statistics of the University of Auckland, Auckland, New Zealand, based on the S language and environment, which was developed at Bell Laboratories by John Chambers and colleagues. Maintained today by a core team — R Core Development Team.

- R provides a wide variety of statistical (linear and nonlinear modelling, classical statistical tests, time-series analysis, classification, clustering,...) and graphical techniques, and is highly extensible.

- One of R's strengths is the ease with which well-designed publication-quality plots can be produced, including mathematical symbols and formulae where needed. Great care has been taken over the defaults for the minor design choices in graphics, but the user retains full control.

- R is viewed as an environment within which statistical techniques are implemented. Can be extended (easily) via packages. R is distributed with a set of base packages that can easealy be extended adding other available through the CRAN family of Internet sites covering a very wide range of modern statistics.

Some documentation

- Manuals:
  1. *An Introduction to R*
  2. *R Installation and Administration*
  3. *R Data Import/Export*
  4. *Writing R extensions*
  5. *R Language Definition*
  6. *Sweave User Manual*

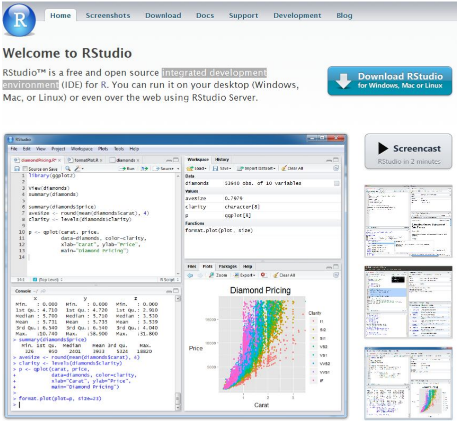
- Other documents freely available:
  1. *Using R for Data Analysis and Graphics - Introduction, Examples and Commentary*, John Maindonald
  2. *Simple R*, John Verzani
  3. *Practical Regression and Anova using R*, Julian Faraway
  4. *An Introduction to R: Software for Statistical Modelling and Computing*, Petra Kuhnert and Bill Venables
  5. *R for Beginners*, Emmanuel Paradis
  6. *Gráficos Estadísticos con R*, Juan Carlos Correa and Nelfi González
  7. *R reference card*, Tom Short
  8. *The R Inferno*, Patrick Burns

- Books: Books related to R
  1. Crawley (2014). *The R book*. John Wily & Sons
  2. Torgo (2009). *A linguagem R - Programação para a análise de dados*. Escolar Editora

RStudio

- R Studio is an *integrated development environment*) to use R;
- R Studio facilitates an introduction to R by providing many shortcuts and convenient features as, e.g., *syntax highlighting, code completion, and smart indentation*;
- From within RStudio you should find (can be customized):
  1. command line (bottom left pane)
  2. code scripts (top left pane)
  3. environment (workspace) objects (top right pane)
  4. file navigator system/plots/help/packages management
- You can customize the aspect of RStudio (e.g. font size and colors of the smart syntax highlighting scheme) via "Tools|Global options".

Website: www.rstudio.org



Welcome to RStudio

RStudio™ is a free and open source integrated development environment (IDE) for R. You can run it on your desktop (Windows, Mac, or Linux) or even over the web using RStudio Server.

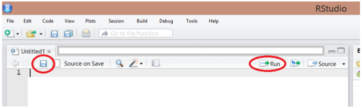
```
library(ggplot2)

View(diamonds)
summary(diamonds)

summary(diamonds$price)
aveSize <- round(mean(diamonds$carat), 4)
clarity <- levels(diamonds$clarity)

p <- qplot(carat, price,
           data-diamonds, color-clarity,
           xlab-"carat", ylab-"Price",
           main-"Diamond Pricing")
```

```
      x               y               z
Min.   : 0.000   Min.   : 0.000   Min.   : 0.000
1st Qu.: 4.710   1st Qu.: 4.720   1st Qu.: 2.910
Median : 5.700   Median : 5.710   Median : 3.530
Mean   : 5.731   Mean   : 5.735   Mean   : 3.539
3rd Qu.: 6.540   3rd Qu.: 6.540   3rd Qu.: 4.040
Max.   :10.740   Max.   :58.900   Max.   :31.800
> summary(diamonds$price)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
    326     950    2401    3933    5324   18820
> aveSize <- round(mean(diamonds$carat), 4)
> clarity <- levels(diamonds$clarity)
> p <- qplot(carat, price,
+            data=diamonds, color=clarity,
+            xlab="Carat", ylab="Price",
+            main="Diamond Pricing")
>
> format.plot(plot=p, size=23)
> 
```

Data	
diamonds	53940 obs. of 10 variables
Values	
avesize	0.7979
clarity	character[8]
p	ggplot[8]
Functions	
format.plot(plot, size)	

R Scripts

- Create a new R script: *File -> New File -> R Script*



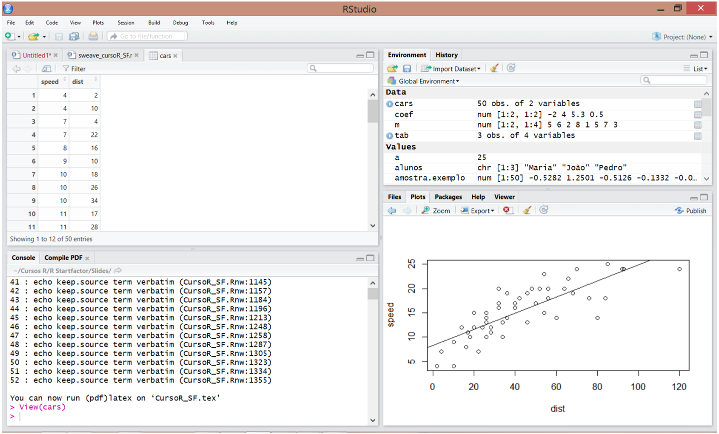
- **All** code should be written here!
- To execute the code (select line or lines): *Run* ou *Crtl+Enter*
- It is advisable to comment all your code. This way you will have a record of everything you did!

```
> 3+3 # This is just to show how to comment your code!
```

- To save your code (file .R): *File -> Save as/Save* or use the upper button.

Tab Environment

- List all objects (vetors, data, functions, etc) created/used in a working session



	speed	dist
1	4	2
2	4	10
3	7	4
4	7	22
5	8	16
6	9	10
7	10	18
8	10	26
9	10	34
10	11	17
11	11	28

Showing 1 to 12 of 50 entries

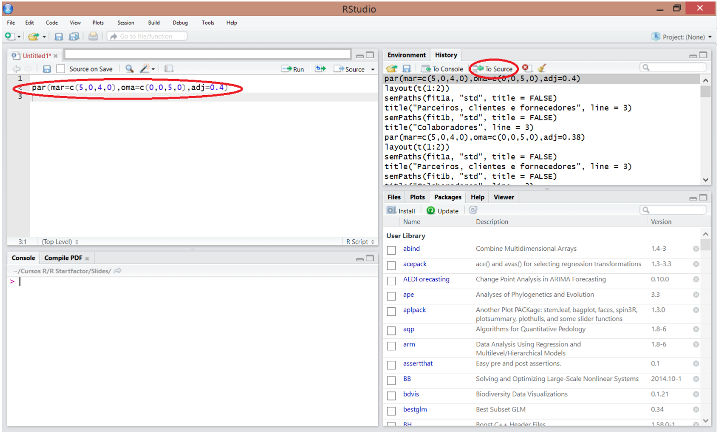
```
Data
cars            50 obs. of 2 variables
coef            num [1:2, 1:2] -2 4 5.3 0.5
m               num [1:2, 1:4] 5 6 2 8 1 5 7 3
tab             3 obs. of 4 variables
Values
a               25
alunos          chr [1:3] "Maria" "João" "Pedro"
amostra.exemplo num [1:50] -0.5282 1.2501 -0.5126 -0.1332 -0.0...
```

```
41 : echo keep.source term verbatim (CursoR_SF.Rnw:1145)
42 : echo keep.source term verbatim (CursoR_SF.Rnw:1157)
43 : echo keep.source term verbatim (CursoR_SF.Rnw:1184)
44 : echo keep.source term verbatim (CursoR_SF.Rnw:1196)
45 : echo keep.source term verbatim (CursoR_SF.Rnw:1213)
46 : echo keep.source term verbatim (CursoR_SF.Rnw:1248)
47 : echo keep.source term verbatim (CursoR_SF.Rnw:1258)
48 : echo keep.source term verbatim (CursoR_SF.Rnw:1287)
49 : echo keep.source term verbatim (CursoR_SF.Rnw:1305)
50 : echo keep.source term verbatim (CursoR_SF.Rnw:1323)
51 : echo keep.source term verbatim (CursoR_SF.Rnw:1334)
52 : echo keep.source term verbatim (CursoR_SF.Rnw:1355)

You can now run (pdf)latex on 'CursoR_SF.tex'
> View(cars)
>
```

- Allows to edit and visualize datasets
- May be saved (.RData file) to upload latter

Tab History

- Records all executed code



```
par(mar=c(5,0,4,0),oma=c(0,0,5,0),adj=0.4)
```

```
~/Cursos R/R Startfactor/Slides/
> 
```

```
par(mar=c(5,0,4,0),oma=c(0,0,5,0),adj=0.4)
layout(t(1:2))
semPaths(fit1a, "std", title = FALSE)
title("Parceiros, clientes e fornecedores", line = 3)
semPaths(fit1b, "std", title = FALSE)
title("Colaboradores", line = 3)
par(mar=c(5,0,4,0),oma=c(0,0,5,0),adj=0.38)
layout(t(1:2))
semPaths(fit1a, "std", title = FALSE)
title("Parceiros, clientes e fornecedores", line = 3)
semPaths(fit1b, "std", title = FALSE)
title("Colaboradores", line = 3)
```

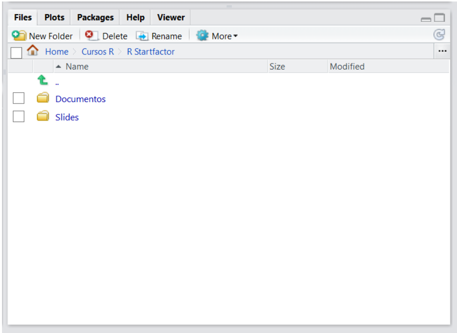
	Name	Description	Version
User Library			
□	abind	Combine Multidimensional Arrays	1.4-3
□	acepack	ace() and avas() for selecting regression transformations	1.3-3.3
□	AEDForecasting	Change Point Analysis in ARIMA Forecasting	0.10.0
□	ape	Analyses of Phylogenetics and Evolution	3.3
□	aplpack	Another Plot PACKage: stem.leaf, bagplot, faces, spin3R, plotsummary, plothulls, and some slider functions	1.3.0
□	aqp	Algorithms for Quantitative Pedology	1.8-6
□	arm	Data Analysis Using Regression and Multilevel/Hierarchical Models	1.8-6
□	assertthat	Easy pre and post assertions.	0.1
□	BB	Solving and Optimizing Large-Scale Nonlinear Systems	2014.10-1
□	bdvis	Biodiversity Data Visualizations	0.1.21
□	bestglm	Best Subset GLM	0.34
□	BH	Boost C++ Header Files	1.58.0-1

- Includes shortcut (*To Source*) to copy command line to R script or to R Console

Tab Files

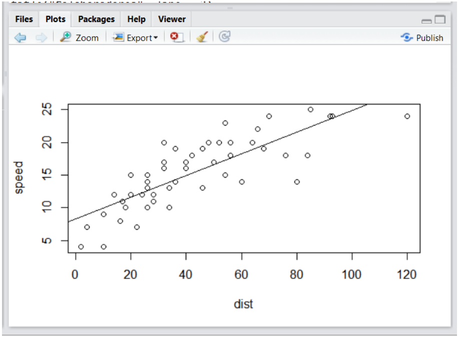
- Allows to manage all files and working folders;

- To define a working directory: *More -> Set as working directory*



Tab Plots

- Graphical display, with navigation arrows



Tab Packages

- All R functions and datasets are stored in *packages*. Only when a package is loaded are its contents available. This is done both for efficiency (the full list would take more memory and would take longer to search than a subset), and to aid package developers, who are protected from name clashes with other code;

- There are about 25 packages supplied with R (called ”standard” and ”recommended” packages) and many more are available through the CRAN family of Internet sites (via https://CRAN.R-project.org) and elsewhere;

- Users connected to the Internet can use the button *install* or the function `install.packages()` to install a new package

```
> install.packages(nortest)
```

- To load a particular package use `library()`.

```
> library(nortest)
```

Tab Help

- R has an inbuilt help facility. To get more information on any specific named function use

```
> ?glm
```

- For searching the help system for documentation matching a given character string, use help.search()

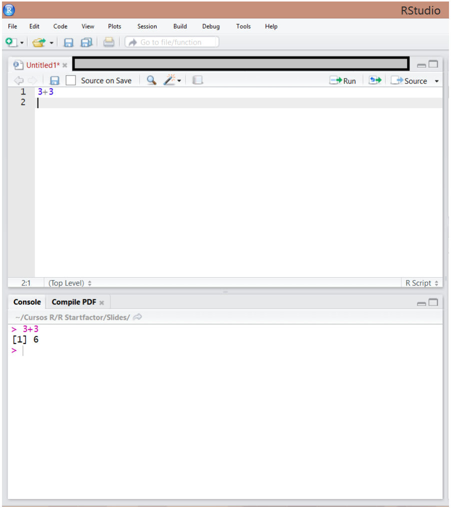
```
> help.search("median")
```

- apropos() and find() give, respectively, a character vector giving the names of objects in the search list matching and where objects of a given name can be found.

```
> apropos("median")
[1] "median"         "median.default" "median_hilow"
> find("median")
[1] "package:stats"
```

Console

- Where executed code appears



```
3+3
```

```
> 3+3
[1] 6
>
```

- It shows the prompt (>) followed by the executed code
- If a command is not complete at the end of a line, R will give a different prompt, by default (+)

R language

R objects

- The entities that R creates and manipulates are known as objects. These may be variables, arrays of numbers, character strings, functions, or more general structures built from such components.
- During an R session, objects are created and stored by name.
- To create an object R typically uses the assignment operator <-, "pointing" to the object receiving the value of the expression, i.e., `object_name <- content`. In most contexts the "=" operator can be used as an alternative. Assignment can also be made using the function `assign()`

```
> a <- 25
> b <- 5
> Total <- a+b
```

- To display an object content, just execute the correspondent line code

```
> Total
[1] 30
```

- To list all created objects of objects: `ls()` ou `objects()`.

```
> ls()
[1] "a"     "b"     "Total"
```

- Removing objects: `rm()`. To remove all objects use `rm(list = ls())`

Types of objects

Includes:

- **Vectors**: The simplest R structure is the numeric vector, which is a single entity consisting of an ordered collection of numbers.
- **Arrays** (caso particular, **Matrices**): An array can be considered as a multiply subscripted collection of data entries, for example numeric (e.g., matrices).
- **Factors**: Vector object used to specify a qualitative feature.
- **Lists**: Ordered collection of objects.
- **Dataframes**: A data frame may be regarded as a matrix with columns possibly of differing modes and attributes.
- **Functions**: Organized structures that are stored in a special internal form and may be used latter. Either "Built-in" or "User-defined".

Tipos de objetos (continuação)

Functions structure:

```
function_name(arg1 = value1, arg2 = value2,...)
```

- Arguments (if several) are comma separated;
- Arguments may be explicitly defined by their names:

```
> rep(x = 5, times = 3)
```

or order:

```
> rep(5,3)
```

- Explicit names allows to ignore order.
- Not using names requires to respect the arguments order.

Object attributes

Mode Defines the basic type of its fundamental constituents including, namely, *numeric*, *complex*, *logical* and *character* modes. To see mode type, use `mode()`

```
> mode(Total)
[1] "numeric"
> c<-"Grupo controlo"
> mode(c)
[1] "character"
```

Class A special attribute known as the class of the object is used to allow for an object-oriented style. Includes all modes and, e.g., classes *matrix*, *dataframe*, *factor*, *array*, *list*. To see the class of an object, use `class()`

```
> class(Total)
[1] "numeric"
```

Length Length of any defined structure: `length()` for vectors/factors and `dim()` for arrays/matrices

```
> length(Total)
[1] 1
```

Vectors

- Function c() allows to create vectors

```
> x<-c(7, 3.1, 8, 18, 12.5)
> x
[1]  7.0  3.1  8.0 18.0 12.5
```

Some particular cases:

- Sequences: Function seq(from,to) and operator from:to.

```
> seq(1,5)
[1] 1 2 3 4 5
> 1:5
[1] 1 2 3 4 5
```

- Repetitions: rep(x,times)

```
> rep(3,4)
[1] 3 3 3 3
```

Matrices

- To create matrices, use `matrix(x,nrow,ncol)`

```
> m<-matrix(c(5,6,2,8),2,2)
> m
     [,1] [,2]
[1,]    5    2
[2,]    6    8
```

- R contains many operators and functions that are available only for matrices:

```
> n<-matrix(c(1,2,3,4),2,2)
> t(m) #transpose
> t(m)%*%n #product
> crossprod(m,n)
> diag(m) #diagonal
```

- Combining matrices : `rbind()` e `cbind()`

```
> cbind(m,m)
     [,1] [,2] [,3] [,4]
[1,]    5    2    5    2
[2,]    6    8    6    8
```

Factors

- Function factor(x) allows R to recognize data as qualitative

```
> f<-factor(c("m","m","m","f","m","f"))
> f
[1] m m m f m f
Levels: f m
```

- Alternatives:

```
> factor(c(1,1,1,0,1,0),labels = c("f","m"))
[1] m m m f m f
Levels: f m
> factor(c(1,1,1,0,1,0),labels = c("m","f"),levels = c(1,0))
[1] m m m f m f
Levels: m f
```

Dataframes

- To create dataframes use data.frame()

```
> stud<-c("Mary","Monica","Petter")
> tab<-data.frame(age=c(5,6,7),name=stud,prof=rep("P1",3))
> tab

  age   name prof
1   5   Mary   P1
2   6 Monica   P1
3   7 Petter   P1
```

- To make the components of a list or a data frame temporarily visible as variables under their component name, without the need to quote the list name explicitly each time, use attach() (CAUTION!)

```
> attach(tab)
> age

[1] 5 6 7
```

- To detach a data frame, use the function detach()

```
> detach(tab)
```

- Rows and columns may also be extracted using matrix indexing conventions (to see latter)

Lists

- Function list() creates lists. There is no particular need for the components to be of the same mode or type:

```
> l<-list(age=c(5,6,7),name=stud,prof="P1")
> l

$age
[1] 5 6 7

$name
[1] "Mary"   "Monica" "Petter"

$prof
[1] "P1"
```

```
> mylist <- list (a = 1:5, b = "Study 1", c = matrix(c(2,4,8,9),2,2))
> mylist

$a
[1] 1 2 3 4 5

$b
[1] "Study 1"

$c
     [,1] [,2]
[1,]    2    8
[2,]    4    9
```

Operatores

Basic operators:

- **Arithmetic**: + - * / ^
- **Relational**: > >= < <= == !=
- **Logical**: !(negation) &(and) |(or)
- **Model formulation**: ∽
- **Indexation**: $
- **Sequence**: :
- **Assignment**: -> <- =

Alguma notação:

- Infinite: $\infty =$ Inf; $-\infty =$ -Inf.
- *Not a number*: $\infty/\infty =$ NaN.
- *Missing values*: NA (*Not Available*).

```
> w<-c(7, 3.1, 8, 18, NA)
```

A função is.na(x) allows to locate NA values.

```
> is.na(w)
[1] FALSE FALSE FALSE FALSE  TRUE
```

Indexation

Index vectors

- `x[n]`
- `x[-n]`
- `x[1:n]`
- `x[c(1,2)]`
- `x[x>2 & x<4]`
- `x[x %in% c(1:5)]`

Index matrices/dataframes

- `x[i,j]`
- `x[i,]`
- `x[,j]`
- `x[c(1,3),]`

Index dataframes

- `dataset$x`: column `x` in `dataset`

Index lists

- `x[[n]]`

Transform thetype of object

- To identify the objects' type and mode use `is.type()` e `is.mode()`
- To change it use `as.type()` e `as.mode()`

Type	Function is.type()	Function as.type()
Array	is.array()	as.array()
Dataframe	is.data.frame()	as.data.frame()
Factor	is.factor()	as.factor()
List	is.list()	as.list()
Matrix	is.matrix()	as.matrix()
Vector	is.vector()	as.vector()
Mode	**Função is.mode()**	**Função as.mode()**
Character	is.character()	as.character()
Complex	is.complex()	as.complex()
Logical	is.logical()	as.logical()
Numeric	is.numeric()	as.numeric()

```
> is.matrix(m)
[1] TRUE
> as.vector(m)
[1] 5 6 2 8
```

```
> is.factor(f)
[1] TRUE
> as.numeric(f)
[1] 2 2 2 1 2 1
```

Math and Statistical generic functions

- Some math functions: `sum(x)`, `sqrt(x)`, `log(x)`, `log(x,n)` , `exp(x)`, `choose(n,x)`, `rank(x)`, `factorial(x)`, `floor(x)`, `ceiling(x)`, `round(x, digits)`, `abs(x)`, `cos(x)`, `sin(x)`, `tan(x)`, `acos(x)`, `acosh(x)`, `gamma(x)`

```
> floor(3.5)
[1] 3
> ceiling(3.5)
[1] 4
```

- Some statistical functions: `max(x)`, `min(x)`, `mean(x)`, `median(x)`, `range(x)`, `var(x)`, `cor(x,y)`, `quantile(x)`, `cumsum(x)`, `cumprod(x)`, `cummax(x)`, `cummin(x)`

```
> y<-c(3.4,6.9,9.4,5.1,3.6)
> cummax(y)
[1] 3.4 6.9 9.4 9.4 9.4
```

Family apply

- This function returns a vector/array/list of values obtained by applying a function to margins of an array or a matrix.

```
> apropos("apply")
 [1] ".rs.api.applyTheme" ".rs.applyTheme"     ".rs.applyTransform"
 [4] "apply"              "dendrapply"         "eapply"
 [7] "kernapply"          "lapply"             "mapply"
[10] "rapply"             "sapply"             "tapply"
[13] "vapply"
```

- Simple example:

```
> apply(m,1,mean)
[1] 3.5 7.0
> apply(m,1,sd)
[1] 2.121320 1.414214
> apply(m,2,sum)
[1] 11 10
```

Other generic functions

- Evaluate an R expression in an specific dataset, without modifying the original data: `with()`
- Evaluate an R expression in an specific dataset, modifying the original data: `within()`
- Check first and last element within an object: `head()` e `tail()`

```
> #example use of with and within
> data(cars)
> head(cars)

  speed dist
1     4    2
2     4   10
3     7    4
4     7   22
5     8   16
6     9   10

> with(cars,mean(speed))

[1] 15.4

> temp<-within(cars,assign("ratio",speed/dist))
> head(temp,4)

  speed dist     ratio
1     4    2 2.0000000
2     4   10 0.4000000
3     7    4 1.7500000
4     7   22 0.3181818

> rm(temp)
```

Probability distributions

Some available distributions:

Distribution	R name	Arguments
Beta	`beta`	`shape1, shape2`
Binomial	`binom`	`size, prob`
Binomial negativa	`nbinom`	`size, prob`
Cauchy	`cauchy`	`location, scale`
Exponencial	`exp`	`rate`
F-Snedecor	`f`	`df1, df2`
Gama	`gamma`	`shape, scale`
Geométrica	`geom`	`prob`
Hipergeométrica	`hyper`	`m, n, k`
Log-normal	`lnorm`	`meanlog, sdlog`
Logística	`logis`	`location, scale`
Normal	`norm`	`mean, sd`
Poisson	`pois`	`lambda`
Qui-quadrado	`chisq`	`df`
t-Student	`t`	`df`
Uniforme	`unif`	`min, max`
Weibull	`weibull`	`shape, scale`

Probability distributions

There are 4 generic functions applicable to probability distributions:

- **probability density function**: `dname(x, ...)`

  Example: $X \frown N(8.5, 2.3) \Rightarrow P(X = 10) = ?$

```
> dnorm(9,8.5,2.3)
[1] 0.1694026
```

- **cumulative density function**: `pname(q, ...)`

  Example: $X \frown N(8.5, 2.3) \Rightarrow P(X \leq 10) = ?$

```
> pnorm(10,mean=8.5,sd=2.3)
[1] 0.7428555
```

Probability distributions

- **qunatile function**: `qname(p, ...)`

Example: $X \frown N(8.5, 2.3) \Rightarrow P(X \leq ?) = 0.743$

```
> qnorm(0.743,8.5,2.3)
[1] 10.00103
```

- **Random**: `rname(n,...)`

```
> rn<-rnorm(1000)
```

Conditional and repetitive execution

- **Conditional execution**: if e ifelse

```
if(contition) expr1 else expr2
```

```
> grades<-c(12,5.3,15,7.0,17)
> if(grades[1]<9.5) print("r") else print("a")
[1] "a"
```

```
ifelse(condition,expr1,expr2)
```

```
> ifelse(grades<9.5,"failed","approved")
[1] "approved" "failed"   "approved" "failed"   "approved"
```

- **Repetitive execution**: for, while e repeat . Instrução break.

```
for (name in expr1) expr2
```

```
> for(x in c(4,9,16,25)) print(sqrt(x))
[1] 2
[1] 3
[1] 4
[1] 5
```

Conditional and repetitive execution

`while (condição) expr`

```
> a <- 0; b <- 1
> while (b < 4) {
 print(b)
 temp <- a + b
 a <- b
 b <- temp
 }
[1] 1
[1] 1
[1] 2
[1] 3
```

`repeat expr`
`break`

```
> x<-1
> repeat{print(x)
 x = x+1
 if(x == 4) {break}}
[1] 1
[1] 2
[1] 3
```

Functions

- To built a function use the following syntax:
  `name <- function(arg1,arg2,...) {expressão}`

- *Example*: $\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$

```
> mymean0<-function(x){sum(x)/length(x)}
> y
[1] 3.4 6.9 9.4 5.1 3.6
> mymean0(y)
[1] 5.68
```

```
> mymean1<-function(x){
 s<-sum(x)
 n<-length(x)
 m<-s/n
 return(m)
 }
```

User-defined functions

- Challenge - **Gaussian density function**:

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma}\, e^{-\frac{1}{2}(\frac{x-\mu}{\sigma})^2}, \; -\infty < x < +\infty, \; -\infty < \mu < +\infty, \; 0 < \sigma < +\infty$$

```
> gaussian1<-function(x,m,s){
   (1/(sqrt(2*pi)*s))*exp(-0.5*((x-m)/s)^2)
 }
> gaussian1(0,0,1)
[1] 0.3989423
```

```
> gaussian2<-function(x,m,s){
   z<-(x-m)/s
   c<-(1/(sqrt(2*pi)*s))
   p<- -0.5*z^2
   g<-c*exp(p)
   return(g)
 }
```

```
> gaussian2(0,0,1)
[1] 0.3989423
> dnorm(0,0,1)
[1] 0.3989423
```

Reading data from files

- Large data objects will usually be read as values from external files rather than entered during an R session at the keyboard.

- To read an entire data frame directly, the external file will normally have a special form:

  1. Columns - variables; Rows - objects;
  2. The first line of the file should have a name for each variable in the data frame;
  3. Missing data: `NA` or blank;

- There are specific packages to specific data formats:

  1. `heaven`: SPSS, Stata, and SAS;
  2. `readxl`: Excel (`.xlsx` ou `.xls`).

Tibbles

- Reading external data using functions from the readxl package creates ”tibbles”
- *Tibbles* are dataframes simplified: they do less (i.e. they don’t change variable names or types, and don’t do partial matching) and complain more (e.g. when a variable does not exist).

```
> library(readxl)
> data<-read_excel("imc.xlsx",sheet = 1,col_names = T)
> data

# A tibble: 540 x 7
   escola idade sexo    imc pabdom panca mgorda
    <dbl> <dbl> <chr> <dbl>  <dbl> <dbl>  <dbl>
 1      2     7 F      15.8     59    67   14.9
 2      2     6 F      13.8     50    60   11.1
 3      2     7 M      16.3     62    69   16.5
 4      2     6 M      17.1     62    70   21
 5      2     7 F      20.2     70    79   30.2
 6      2     7 F      13.3     50    61   11.4
 7      2     7 M      14.6     56    62   14.9
 8      2     6 M      15.2     53    65   16.4
 9      2     7 F      18.8     65    67   26.7
10      2     6 F      15.6     53    67   14.3
# ... with 530 more rows
```

- tibble() does much less than data.frame(). We will transform it to dataframes (easier).
- There are specific functions to tibbles (not covered here)

Example

```
> data<-as.data.frame(data)
```

```
> data[1:6,]
  escola idade sexo   imc pabdom panca mgorda
1      2     7    F 15.78     59    67   14.9
2      2     6    F 13.84     50    60   11.1
3      2     7    M 16.27     62    69   16.5
4      2     6    M 17.10     62    70   21.0
5      2     7    F 20.18     70    79   30.2
6      2     7    F 13.26     50    61   11.4
> data[1:6, 4]
[1] 15.78 13.84 16.27 17.10 20.18 13.26
> data[1:6, "panca"]
[1] 67 60 69 70 79 61
> mean(data$imc)
[1] 17.2073
```

Summarizing datasets

- Dimensions: dim() or nrow() and ncol()

```
> dim(data)
[1] 540   7
```

- To modify a numeric variable to a qualitative variable, use factor()

```
> data$escola <- factor(data$escola, labels=c("EscA","EscB","EscC","EscD"))
```

- Function str() summarizes the dataframe structure

```
> str(data)
'data.frame':        540 obs. of  7 variables:
 $ escola: Factor w/ 4 levels "EscA","EscB",..: 2 2 2 2 2 2 2 2 2 2 ...
 $ idade : num  7 6 7 6 7 7 7 6 7 6 ...
 $ sexo  : chr  "F" "F" "M" "M" ...
 $ imc   : num  15.8 13.8 16.3 17.1 20.2 ...
 $ pabdom: num  59 50 62 62 70 50 56 53 65 53 ...
 $ panca : num  67 60 69 70 79 61 62 65 67 67 ...
 $ mgorda: num  14.9 11.1 16.5 21 30.2 11.4 14.9 16.4 26.7 14.3 ...
```

Data selection/filtering

- To select a subset, use `subset()`

```
> EscolaA<-subset(data,escola=="EscA")
```

- You can also use indexation to filter the dataset:

```
> data[data$escola=="EscA",c(3,4)]
```

```
> data[data$imc>25 & data$sexo=="M",]

    escola idade sexo   imc pabdom panca mgorda
529   EscC     9    M 25.52     84    93   28.2
540   EscC     9    M 25.08     79    85   29.8
```

- To randomly select a sub-sample, use `sample()`

```
> n.amostra<-4
> data[sample(1:nrow(data),n.amostra),]

    escola idade sexo   imc pabdom panca mgorda
178   EscA     6    F 16.05     56    59   20.3
341   EscC     9    M 15.36     56    63   15.8
305   EscC     7    F 16.95     57    71   22.7
499   EscC    10    F 18.76     51    69   25.2
```

Data ordering

- To order a dataset based on the values of a particular variable (or variables) use `order()` (ascending) or `rev()` and `order()` (descending) (ordenação decrescente)

```
> data[order(data$idade),]
> data[rev(order(data$idade)),]
```

- To order a dataset based on the values of two (or more) variables:

```
> data[1:5,]

  escola idade sexo   imc pabdom panca mgorda
1   EscB     7    F 15.78     59    67   14.9
2   EscB     6    F 13.84     50    60   11.1
3   EscB     7    M 16.27     62    69   16.5
4   EscB     6    M 17.10     62    70   21.0
5   EscB     7    F 20.18     70    79   30.2

> data_ord<-data[order(data$escola,data$idade),]
> data_ord[1:5,]

    escola idade sexo   imc pabdom panca mgorda
160   EscA     6    F 16.88     58    63   21.5
163   EscA     6    M 16.00     57    64   17.8
164   EscA     6    M 20.60     70    74   25.7
165   EscA     6    M 17.51     60    63   19.8
166   EscA     6    M 15.75     53    58   19.2
```

Introducing dplyr

- dplyr is a grammar of data manipulation, providing a consistent set of verbs that help you solve the most common data manipulation challenges:
    - mutate() adds new variables that are functions of existing variables
    - select() picks variables based on their names.
    - filter() picks cases based on their values.
    - summarise() reduces multiple values down to a single summary.
    - arrange() changes the ordering of the rows.
- All these can be combine with group_by() to perform any operation ”by group”.
- It uses the pipe operator %>%

```
> library(dplyr)
> data %>%
   filter(escola == "EscD")
  escola idade sexo   imc pabdom panca mgorda
1   EscD     6    F 15.53     55    63   19.4
2   EscD     7    F 16.88     56    69   18.2
3   EscD     6    F 15.78     54    61   19.8
4   EscD     7    M 14.28     53    65   12.6
5   EscD     6    F 17.87     57    69   22.2
6   EscD     6    M 15.99     56    66   16.9
7   EscD     6    M 14.19     50    60   14.4
```

Introducing dplyr

```
> data %>% mutate(escola, MG = mgorda/100) %>% filter(escola == "EscD")

  escola idade sexo   imc pabdom panca mgorda    MG
1   EscD     6    F 15.53     55    63   19.4 0.194
2   EscD     7    F 16.88     56    69   18.2 0.182
3   EscD     6    F 15.78     54    61   19.8 0.198
4   EscD     7    M 14.28     53    65   12.6 0.126
5   EscD     6    F 17.87     57    69   22.2 0.222
6   EscD     6    M 15.99     56    66   16.9 0.169
7   EscD     6    M 14.19     50    60   14.4 0.144
```

```
> data %>% group_by(sexo) %>% summarise(n = n(), imc = mean(imc, na.rm = TRUE))

# A tibble: 2 x 3
  sexo      n   imc
  <chr> <int> <dbl>
1 F       270  17.4
2 M       270  17.1
```

```
> data %>% filter(escola == "EscD") %>% arrange(desc(imc))

  escola idade sexo   imc pabdom panca mgorda
1   EscD     6    F 17.87     57    69   22.2
2   EscD     7    F 16.88     56    69   18.2
3   EscD     6    M 15.99     56    66   16.9
4   EscD     6    F 15.78     54    61   19.8
5   EscD     6    F 15.53     55    63   19.4
6   EscD     7    M 14.28     53    65   12.6
7   EscD     6    M 14.19     50    60   14.4
```

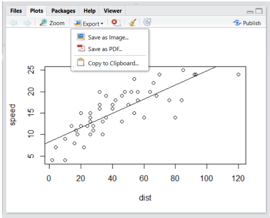
Graphical systems

Graphical systems

- R has several graphical systems:
  1. **base** (package `graphics`)
  2. **lattice** (packages `lattice`)
  3. **ggplot2** (packages `ggplot2`)

- Plotting commands are divided into three basic groups::
  1. **High-level**: plotting functions create a new plot on the graphics device, possibly with axes, labels, titles and so on
  2. **Low-level**: plotting functions add more information to an existing plot, such as extra points, lines and labels
  3. **Interactive**: graphics functions allow you interactively add information to, or extract information from, an existing plot, using a pointing device such as a mouse

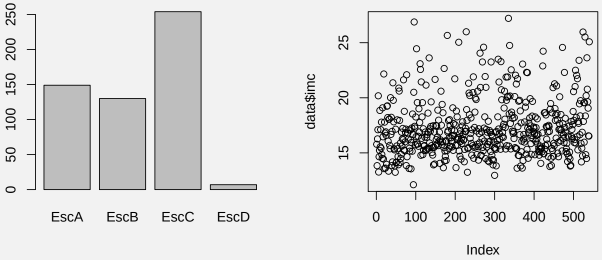
- Possible formats include: pdf, png, jpg, eps, ps,...



base graphical system

- One of the most frequently used plotting functions in R is the plot() function. This is a generic function: the type of plot produced is dependent on the type or class of the first argument.

```
> plot(data$escola)
> plot(data$imc)
```

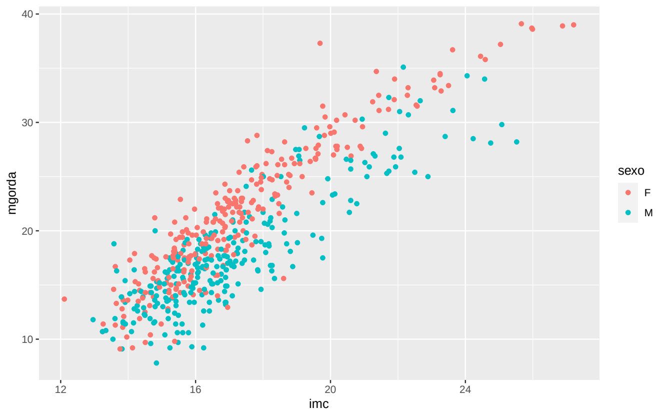


- Other high level plot functions include: boxplot(), barplot(), hist(), pie(), qqnorm(), qqplot(), curve(), ... Tip: apropos("plot") shows many other.

ggplot2 graphical system

- ggplot2 is a system for declaratively creating graphics, based on *The Grammar of Graphics*. You provide the data, tell ggplot2 how to map variables to aesthetics, what graphical primitives to use, and it takes care of the details. Basic function is `ggplot()`

```
> library(ggplot2)
> ggplot(data, aes(imc, mgorda, colour = sexo)) +
   geom_point()
```


mgorda
imc
sexo
● F
● M