Mathematical Models and Applications (21/22)

Module I - Reviewing Basic Probability and Statistics concepts with the R software

Module I – classes: - 1 to 16 March 2022

Manuela Neves

ISA/ULisboa

Lesson 1–Outline

Module I – Program

- Brief introduction to the R environment.
- The main objects in R. Operations with numbers, vectors and matrices.
- Structure and manipulation of data in R. Some statistical functions.
- Descriptive Statistics and Exploratory Data Analysis. Visualization of data in one and two dimensions.
- The main discrete and continuous probability models.
- Theory of Estimation: introduction. The estimator and the estimate notion.
- Point Estimation methods: the method of moments and the method of maximum likelihood.
- Introduction to parametric and non-parametric statistical inference.
- Confidence Intervals and Hypothesis Testing.
- Exercises.

R tutorials

- Everitt, B.S. and Hothorn, T. (2006). *A Handbook of Statistical Analyses using R*. Chapman & Hall
- Kerns, G.J. (2010). *Introduction to Probability and Statistics using R.* Disponível *on-line*
- Monteiro, L.R. (2006). *Introdução à Biometria usando o R*. Disponível *on-line*.
- Torgo, L. (2006). *Introdução à Programação em R*. Disponível *on-line*
- Verzani, J. (2002). *Using R for Introductory Statistics*. Disponível *on-line*

References

- Casella, G. and Berger, R.L.(2002). *Statistical Inference*. Wadsworth & Brooks
- Murteira, B. e Antunes, M. (2012). *Probabilidades e Estatística. VolI e II.* McGraw-Hill
- Neves, M. M. (2017). *Introdução à Estatística e à Probabilidade com com utilização do ISAPress*.

LISBOA
INSTITUTO SUPERIOR DE AGRONOMIA
INTRODUÇÃO
à ESTATÍSTICA e à PROBABILIDADE
com utilização do
Manuela Neves

- Pestana, D. e Velosa, S. (2008). *Introdução à Probabilidade e à Estatística.* Fundação Calouste Gulbenkian

LESSON 1

Brief introduction to the R environment.

A brief introduction to the R environment

- **What is R and why R?**
  - It is an integrated set of computational tools that allow performing statistical analysis, numerical calculation and has excellent graphing capabilities.
  - It contains advanced statistical routines not yet available in other packages.
  - It is an interpreted language – the commands are immediately executed.
  - It is a language oriented by objects - the data are stored in the active memory of the computer in the form of objects, have name and actions are applied over them.

A brief introduction to the R environment (cont.)

- **What is R ?**
  - It is a computer programming language, it allows to implement and to treat new algorithms.
  - It is constantly updated by the introduction of new and diverse statistical procedures.
  - It is a free public-key distribution application available in (http://cran.r-project.org/) - - here is all the information.
  - After doing the download of the appropriate version computer operating system (eg. R-4.0.2- it appears in the desktop Rx64 4.02).

A brief introduction to the R environment(cont.)

- **Advantages of R ?**
  - It's totally free.
  - It is the result of an international collaboration of several researchers, who maintain a network of *internett*.
  - It is possible to more easily fix the *bugs* detected and even get help to try to solve something more specific.
  - In the *homepage* R, `www.r-project.org` there are several tutorials in various languages.
  - It is possible to develop a module (*package*) for an application of interest, make it available in R so that you can share knowledge.
  - The R environment has several *mailing lists* for a wide variety of topics, see the *homepage* R.

A brief introduction to the R environment (cont.)

- **Starting and ending a session in R**
  - Create a folder (eg MMA.20.21) where all working files will be stored: data files, output files, R files...
  - Start the R – a console window will open.
  - Specify the workspace
    menu: File $\longrightarrow$ Change dir ...
  - The commands are given after the `prompt >` and they are executed after pressing **“Enter”**.

A brief introduction to the R environment (cont.)

- **Starting and ending a session in R**
  - To check if you are in the folder you can write
    `> getwd()`
  - To end a session run `>q()`.
  - If you want to save the *workspace* (work session that contains the set of working objects) do **Yes** and the session is saved in the file **.Rdata**

Dealing with *packages*

- All the functions and commands in R are stored in *packages*.
  - **To:**
    - see which packages are available `>(.packages())`
    - see which packages are installed `>library()`
    - load an installed package into memory
      `>library(name-package)` or
      menu: Packages $\longrightarrow$ Load Package ...
- To install a *package* do
  menu: Packages $\longrightarrow$ Install Package ...
- In an R session the contents of a package are only available when it is loaded into memory.

Getting help in R

- About a *package*
  `>help(package=datasets)`
- About a dataset
  `>help(InsectSprays)` or `> ?InsectSprays`
- About a function when we know the name
  `>help(mean)` or `> ?mean`
- To search for a string of characters
  `>help.search("norm")` or `>??"norm"`
  it displays the package and command where the sequence appears

```
stats::Normal                 The Normal Distribution
```

The R as a calculator:

- Arithmetic expressions (here the result is shown and not saved):

```
> 2 + 3/4 * 7^2
[1] 38.75
> exp(-2)/log(sqrt(2))
[1] 0.3904951
> sin(pi)^2 + cos(pi)^2
[1] 1
>sin((pi)^2) + cos((pi)^2)
[1] -1.332987   # See the difference!!!!
```

- Assignment (result saved in object):

```
> x<-  3    # the result is saved in the variable x
> x       # display the contents of x
```

Some of the most usual functions in the R:

- `sqrt()` - square root
- `abs()` - absolute value
- `log()` - natural logarithm
- `log10()` - base 10 logarithm
- `exp()` - exponential
- `sin()` ; `cos()` ; `tan()`
- `factorial(`n`)` - factorial, $n!$
- `choose(n,k)` - gives $\binom{n}{k}$

Dealing with objects in R

- R objects are entities that R creates, manipulates, and can be stored in a *workspace*.
- To view the list of objects in the *workspace*: `> ls()`
- To view the information about objects in the *workspace*:
  `> ls.str()`
- To delete objects: `> rm(x, y)`
- To delete **all** the existing objects in the *workspace*
  `> rm(list=ls())`
- To save the *workspace* in a file: `> save.image()` or `menu: File⟶Save Workspace ...`
- The file *workspace* by default is `.RData`

How to better handle with R - use of *script*

Instead of typing commands directly in the R console, they can be written and stored in `text files` now without errors and even commented, to facilitate its later use.
These files must have a **.R** extension and must be saved in the folder.

To:

- Create a *script* file
  `menu:  File⟶ New script ...`
- Use a *script* file
  `menu:  File⟶ Open script ...`

Structure and manipulation of data in R

Objects in R

- The **Objects** in R are characterized by:
  - name;
  - type - ex. vector, matrix, factor, array, data frame, ts, list, function;
  - attributes:
    –*mode*: numeric, character, complex, logical;
    –*length*: number of elements in the object;

  `>str(x)` - shows the object x internal structureo
- **Name**
  - Must start with a letter (A-Z ou a-z);
  - May contain digits and/or dots;
  - *Case-sensitive*
  - *Names to avoid* (because they are used internally by R).
    Examples:

    c, q, t, C, D, F, I, T, pi, diff, df, pt, if, else, for, in, next, repeat, else, while, break, NULL, NA, NaN, Inf, FALSE, TRUE

Objects in R - `Vector`

Vector: data structure of the same type (numeric or characters) – is the most common object type.

- How to create a vector - with `c()`

  ```
  > x <- c(1.2, 5.7, 6.3, 8, 14)
  > cores <- c("Red","Green","Blue")
  > u <- c(F,T,F)
  > mais.cores <- c(cores, "Yellow","Black")
  ```

- A vector can contain special symbols: `NA` (unknown value, *missing value*), `NaN` *(Not a Number)* , `Inf` , `- Inf`.

  ```
  z <- c(log(0),NA,Inf);z
  [1]   -Inf   NA   Inf
  ```

Objects in R - `Vector`

- Generation of sequences (allows to create certain vectors)

```
> y <- 1:5
> w <- seq(1, 1.4, by = 0.1)
> w1 <- rep(1,7)
> w2 <- rep(1:3,2)
```

- **Operations with vectors**

```
> v1 <- c(1,3,-1,2);    v2 <- c(2,4,5,1)
```

Remark: operations are performed element by element – if one of the vectors is smaller than the other it is concatenated with itself

```
> v1+v2; v1*v2; v1*2; 2/v1
```

Objects in R - `Vector`

- **Logical operators**

  `> x>4; x>4 & x<6` (& conjunction)

  `> x<5 |x >= 8` (| disjunction)

  `> 2==sqrt(4)      [1] TRUE`

- **Selecting elements from a vector** – usa-se [ ]

  `> cores[1]` - returns the 1st component of the vector cores

  `> cores[-c(1,3)]` - shows the resulting vector after removing the elements in positions 1 and 3 of the vector cores

  `> x[u]` - returns the components of x corresponding to the components TRUE of u

  `> x[x>2 & x<14]` - returns the components of x between 2 and 14

R functions on numerical vectors

Some functions performed element by element

> `length(x)` - returns the number of elements of the vector `x`

> `sort(x)` - returns a vector with the elements of the vector `x` ordered in ascending order

> `sum(x)` - returns the sum of the elements of the vector `x`

> `prod(x)` - returns the product of the elements of the vector `x`

> `cumsum(x)` - returns a vector whose elements are the cumulative sum of the elements of vector `x`

> `cumprod(x)` - similar to the previous, with product

> `max(x); min(x)` - returns the maximum and the minimum of the elements of the vector `x`

> `factorial(x)` - returns, for each component x_i, $\Gamma(x_i + 1)$

> `sample(x)` - makes a permutation of the elements of the vector `x`

Objects in R - `Matrix`

A matrix is a data structure, from the same type, referenced by two indexes (two dimensions). It is defined by the number of lines nrow and number of columns ncol and a set of nrow $\times$ ncol values.

```
>M <- matrix(1:12,nrow=3,ncol=4);M
>rownames(M)<-c("L1","L2","L3")
>colnames(M)<-c("C1","C2","C3","C4")
M
```

```
     [,1] [,2] [,3] [,4]          C1  C2  C3  C4
[1,]   1    4    7   10        L1  1  4   7   10
[2,]   2    5    8   11        L2  2  5   8   11
[3,]   3    6    9   12        L3  3  6   9   12
```

Values are set by column by default.

```
> M <- matrix(1:12,3,4,byrow=T)
```

Selecting elements from an array

```
> M[3,1]        # element that is on line 3 and column 1
> M[3,]         # vector with elements from line 3
> M[,1]         # vector with elements from column 1
> M[c(1,4),]    # elements of line 1 and line 4
                #  of all columns
> M[-2,c(1,4)] # submatrix constituted by
                 # lines 1 and 3 and columns 1 and 4
```

Matrices operations

```
> A<-matrix(c(3,2,-4,0,-1,8,2,1,3,4,2,0),nc=4):A
> A * M     # makes the product element by element
            # (see the result!!!)
> A %*% M  # performs the usual matrix product
> t(A)       # transposed from A
> diag(k)    # does the identity matrix of dimension k
> rowMeans(A) # returns the vector of averages per line
> solve(A)     # inverse of A
> solve(A,b)  # returns the vector x in equation Ax=b

> vec <- rep(2,4)
> M * vec      # By replication has the vector (2,2,2,2)
               # it makes the product component to component
> M %*% vec  #makes the usual product (mathematical)
                # of matrix M by vector (2,2,2,2)
```

Objects in R – `List`

- List: an ordered collection of objects (list components) that can be of different types (numeric vectors, logical vectors, matrices, functions,...)

```
> aluno <- list(num=12345, nome="Manuel",
+ notas=c(12.5,13.4,12.1,14.3),curso="Eng.Florestal")
> str(aluno)
```

- The components of a list can be referred to by its index, with [[]] or by its designation

```
> aluno[[2]]         # second component
> aluno$nome         # component named as  "nome"
> aluno[2:3]         # gives a sublist
```

Nota: The result of many functions is a list.

Objects in R – `Data Frame`

A data frame is similar to a matrix in which the columns may contain data of different types.
A `data frame` can be seen as a table with data: columns - are the variables; lines – are the records (the observations in each variable)
It is the usual framework for storing data

Reading a `data frame` existing in R

```
> data()   # show several data frame existing
           # in package  "datasets"
> data(ToothGrowth) # loads the data to
                     # the memory of R
> ToothGrowth # shows the  values contained
                 # in data frame
```

Dealing with a `Data Frame`

```
> str(ToothGrowth)
              # shows the structure of the data frame
> head(ToothGrowth)
              # to display the first 6 lines
> names(ToothGrowth)
              # gives the variable names (columns)
> dim(ToothGrowth)
              # gives the dimension (n.lines, n. columns)
> ToothGrowth[,2]
              # gives rhe 2nd coluna
> ToothGrowth$len
              # shows the values of the variable "len"
```

Dealing with a `Data Frame`

Dealing with a data frame can be simpler by using the function `attach()`.
It allows you to directly access columns of a *data frame*, without referring to the name of the `data frame`.

```
> len                  # unknown object
> attach(ToothGrowth)
> len  # allows you to access the columns of the data frame
> detach (ToothGrowth) # the inverse of the attach operation
> len  # object unknown again
```

A famous `Data Frame`

A famous database is `iris`, existing in R. It provides 4 measures observed in 3 species of *iris* (It is a data frame with 150 rows and 5 columns)

```
>iris

    Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
1            5.1         3.5          1.4         0.2     setosa
2            4.9         3.0          1.4         0.2     setosa
3            4.7         3.2          1.3         0.2     setosa
....

> names(iris)

[1] "Sepal.Length" "Sepal.Width"  "Petal.Length" "Petal.Width"  "Species"

> dim(iris)

[1] 150   5
```

A famous Data Frame

```
> iris[,1]

  [1] 5.1 4.9 4.7 4.6 5.0 5.4 4.6 5.0
  [9] 4.4 4.9 5.4 4.8 4.8 4.3 5.8 5.7 ...

> mean(iris[,1])

[1] 5.843333

> summary(iris[,1])   # o the same as summary(iris$Sepal.Length)

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  4.300   5.100   5.800   5.843   6.400   7.900

> table(iris$Species)

    setosa versicolor  virginica
        50         50         50
```

Creating a `Data Frame`

```
> pauta <- data.frame(N.Aluno = c(18355, 17456, 19334, 17756),
+   turma = c("T1", "T2", "T3", "T3"),
+   notas.Est = c(10.3,9.3, 14.2, 15))

> pauta; pauta$notas.Est

   N.Aluno  turma   notas.Est
1   18355    T1       10.3
2   17456    T2        9.3
3   19334    T3       14.2
4   17756    T3       15.0
...
[1] 10.3  9.3 14.2 15.0
```

Importing data from a `Data Frame`

One of the most common ways to store data to work on R is to use text files.
Having a file in the format `txt` or `dat` or *Comma Separated Values*), i.e., the values in each line are separated by commas or semicolons **must**

1. open the file with a text editor `(Notepad, Wordpad)` to display the structure
2. to read it use the command `read.table()`

```
>read.table("ficheiro",header=TRUE)
```

depending on the data structure.

Reading files

When you have data in Excel each sheet must be saved in a file `csv`. Depending on your computer's settings, the columns will be separated by a comma **(,)** or a semicolon **(;)**

Exemplo

```
>semente<-read.table("sementes.csv",header=TRUE,
 + dec =".",sep=";",as.is = TRUE,na.strings = "NA")
>head(semente)
```

```
  melhorada tradicional
1      3.46        3.18
2      3.48        3.67
3      2.74        2.92
4      2.83        3.10
```

Writing files

There are other reading functions, similar to `read.table ()`, whose differences reside in the `separator` which is used by default

`read.csv()` – the decimal separator is dot;
`read.csv2()` – the decimal separator is comma

Writing files

To write the contents of a `date frame` , "x ", in an excel-compatible "output.csv "file, use the function

```
>write.table(x,file="output.csv",sep=";",
  +  dec=".",row.names=FALSE)
```

To write to a file `.txt` compatible with *Notepad* do

```
>write.table(x,file="output.txt",sep=","
    +  ,row.names=FALSE)
```

Functions and Programming in R

Functions in R

The has a vast set of defined functions object-oriented

Estrutura:

```
>function (compulsory arguments, optional arguments)
```

- Example of standard functions (we have already mentioned some of them)

  ```
  abs() log() log10() sqrt() round(x,3)
  exp() sin() cos() tan() gamma() choose(n,k)
  ```

- Functions in Matrix Algebra

  ```
  t(X)  nrow(X)  eigen(X)  solve(A,b)  det(X)
  ```

Functions in R

- Statistical functions

```
mean()  median()  quantile(x,prob=p)
var()  sd() plot() barplot()
summary()  sample() hist()  boxplot()
predict()  lm()  aov()  t.test()
```

Later on we will see more ...

Creating a function in R

- A function is defined by a name, a list of arguments separated by commas and a block of instructions (function body)

General expression

```
>function(arguments) {
   commands
    }
```

Example 1. The logistic standard model

```
>Flogistica<- function(x){1/(1+exp(-x))}
               # cumulative distribution function
>dlogistica<- function(x){exp(-x)/(1+exp(-x))^2}
               # density function
>Flogistica(0); dlogistica(0)
```

Creating a function in R

Example 2. The coefficient of variation of a data set

```
>coef.var<- function(x) {
    cv<-sd(x)/mean(x)
    return(cv)
     }

>z<-c(2,4,6,2,4,8,9,1,3,2,7,8,3,2)
>coef.var(z)

[1] 0.621123
```

Control structures

The function `for ( )` . Syntax

```
> for (indice  in sequencia) {
  expression to execute
 }
```

Example

```
> x<-c(2,4,6,2,4,8,9,1,3,2,7,8,3,2)
> soma<-0
> for (i in 1:length(x))
      {soma<-soma+x[i]}
> soma
```

Control structures

The structure `if()`

```
>conta<-0;xval<-rnorm(10);xval;soma<-0
>for (i in 1:10)
{
>if (xval[i]<0) {conta<-conta+1}
             else
             {soma<-soma+xval[i]}
}
>conta;soma
>print(conta);print(soma)
```

Descriptive Statistics and Exploratory data analysis

Exploratory data analysis – Example

- Consider the data frame InsectSprays from the *package* datasets in R.

```
>help(InsectSprays)
>data(InsectSprays)
>head(InsectSprays)

>str(InsectSprays)
```

```
'data.frame':   72 obs. of  2 variables:
 $ count: num  10 7 20 14 14 12 10 23 17 20 ...
 $ spray: Factor w/ 6 levels "A","B","C","D",..:...
```

The variable Spray is a factor with 6 levels

Exploratory data analysis –some basic concepts.

Population or Universe and Statistical Unit
Variable: feature of interest
Sample: population subset or observed dataset

The exploratory data analysis aims to: organize, summarize, present and extract information from a data set.

The variables that are of interest may be qualitative (nominal or ordinal) or quantitative (discrete or continuous)

To the data (raw material of Descriptive Statistics) is usually assigned the same classification, i.e. may be qualitative (nominal or ordinal) or quantitative (discrete or continuous)

Exploratory data analysis - some basics

The procedures that Descriptive Statistics can use depend on the nature of the data.

For the data of qualitative nature the usual procedures for their descriptive study are: to use frequency tables, bar diagrams and can be obtained the mode

For the data of quantitative nature one can make its descriptive study with frequency tables, histograms or bar charts and one can calculate a variety of numerical Indicators: the mean, the median and quantiles, the variance and the standard deviation, etc.

Exploratory data analysis–the example again

- Basic descriptive analysis: function `summary()`

```
>summary(InsectSprays)

   count            spray
 Min.    : 0.00    A:12
 1st Qu.: 3.00    B:12
 Median : 7.00    C:12
 Mean    : 9.50    D:12
 3rd Qu.:14.25    E:12
 Max.    :26.00    F:12
```

- Basic descriptive analysis by subgroups: function `by()`

```
> by(InsectSprays$count,InsectSprays$spray,summary)
```

Exploratory data analysis

```
InsectSprays$spray: A
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   7.00   11.50   14.00   14.50   17.75   23.00
------------------------------
InsectSprays$spray: B
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   7.00   12.50   16.50   15.33   17.50   21.00
------------------------------
InsectSprays$spray: C
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  0.000   1.000   1.500   2.083   3.000   7.000
------------------------------
InsectSprays$spray: D
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  2.000   3.750   5.000   4.917   5.000  12.000
...
```

Exploratory data analysis–tables

```
>summary(InsectSprays[InsectSprays$count>10,])
```

```
     count         spray
 Min.    :11.00   A: 9
 1st Qu.:13.00   B:11
 Median :16.00   C: 0
 Mean    :16.68   D: 1
 3rd Qu.:20.00   E: 0
 Max.    :26.00   F:10
```

Alternatively you can use the command `subset()`

```
>summary(subset(InsectSprays,count>10))
```

Exploratory data analysis–tables

Table of frequencies – if we have a qualitative variable or a discrete quantitative with few different values

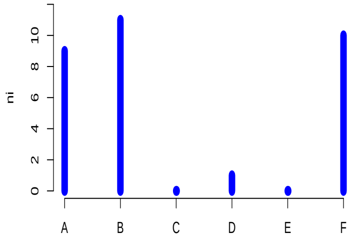
```
> parte.dados<-subset(InsectSprays,count>10)

> head(parte.dados)
  count spray
3    20     A
4    14     A
5    14     A
6    12     A
8    23     A
9    17     A
```

Exploratory data analysis–tables

```
> ni<-table(parte.dados$spray) #freq. absoluta
> fi<-ni/sum(ni)
> fi.ar<-round(fi,3)

> cbind(ni,fi,fi.ar)
  ni         fi  fi.ar
A  9 0.29032258  0.290
B 11 0.35483871  0.355
C  0 0.00000000  0.000
D  1 0.03225806  0.032
E  0 0.00000000  0.000
F 10 0.32258065  0.323
```



Exploratory data analysis – tables

Table of frequencies – quantitative variables

```
> table(InsectSprays$count)

 0  1  2  3  4  5  6  7  9 10 11 12 13 14 15 16 17 19 20 21 22
 2  6  4  8  4  7  3  3  1  3  3  2  4  4  2  2  4  1  2  2  1
```

The variable count of `data frame InsectSprays` is discrete but as we see it presents many different values. In this case, such as with a continuous variable, the construction of a frequency table to summarize the data is obtained using the command `hist()`.

The data will be grouped into classes (use of the Sturges Rule) or classes defined by the user.

```
> ?hist
```

Exploratory data analysis – tables

```
> attach(InsectSprays)
> hist(count,plot=F) #returns a list

$breaks
[1]  0  5 10 15 20 25 30
$counts
[1] 31 10 15  9  5  2
$density
[1] 0.086111111 0.027777778 0.041666667 0.025000000 0.013888889 0.005555556
$mids
[1]  2.5  7.5 12.5 17.5 22.5 27.5
$xname                      $equidist          attr(,"class")
[1] "count"                 [1] TRUE          [1] "histogram"
```

Exploratory data analysis – histograms

The result of the function `hist(count,plot=F)` is then a `List` with the following components

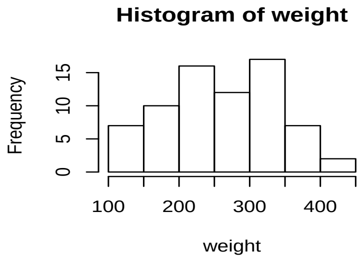
`breaks` - class limits
`counts` - absolute frequency of each class
`density` - relative frequency / (amplitude of each class)
`mids` - midpoint of each class
`equidist` - a logical quantity that indicates whether or not the classes have constant amplitude

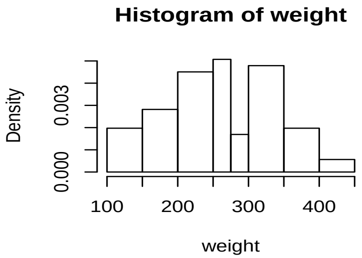
Exploratory data analysis – histograms

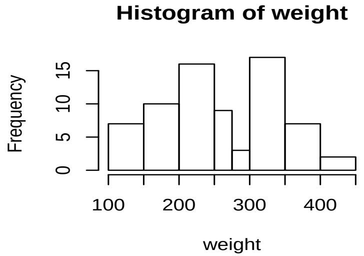
Remark: if the classes have different amplitudes, the height of each rectangle is the relative frequency/(amplitude of the class) - done by default, in R

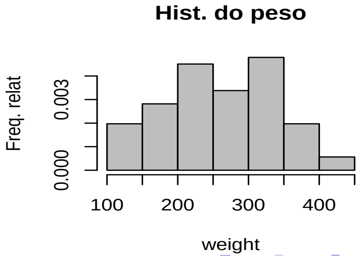
```
> data(chickwts)
> head(chickwts)
> par(mfrow=c(2,2))   # allows to represent 4 graphs
> hist(weight,breaks=
+    c(seq(100,250,50),275,seq(300,450,50)))
+    # compare the heights of classes 3 and 4
> hist(weight, freq=T,breaks=
+    c(seq(100,250,50),275,seq(300,450,50))) $##$message
> hist(weight,col="grey",main="Hist. do peso",
+    freq=F,ylab="Freq. relat")
```

Exploratory data analysis









Exploratory data analysis– some indicators

Shape indicators: skewness and kurtosis coefficients. Need to load the package`fBasics`

```
>library(fBasics)
> x <- c(0:10, 50)
> skewness(x) #coeficient of assymetrie

[1] 2.384115
attr(,"method")
[1] "moment"

>kurtosis(x) #coeficient of kurtosis

[1] 4.586300
attr(,"method")
[1] "excess"
```

Exploratory data analysis

```
>basicStats(x) # from package fBasics

nobs          12.000000
NAs            0.000000
Minimum        0.000000
Maximum       50.000000
1. Quartile    2.750000
3. Quartile    8.250000
Mean           8.750000
Median         5.500000
Sum          105.000000
SE Mean        3.859512
LCL Mean       0.255271
UCL Mean      17.244729
Variance     178.750000
Stdev         13.369742
Skewness       2.384115
Kurtosis       4.586300
```

Graphs in R

R allows a wide variety of graphics, see

```
>demo(graphics)
>demo(persp)
```

Commands for creating charts are divided into two large groups:

- high-level graphical functions - allow to create a new graphic;
- low-level graphical functions - allow to add information to an existing chart.

Graphs in R with plot()

plot(tab) – produces a graph with bars if tab is a table associated with a numerical vector

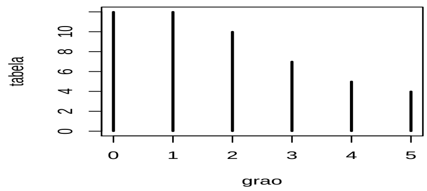
```
>grao<-c(1,2,0,0,1,4,2,5,1,1,5,0,2,2,3,2,1,0,0,3,3,3,
2,2,5,5,0,3,1,0,0,1,1,2,0,4,1,4,0,3,4,2,3,1,1,0,2,0,4,1)
>tabela<-table(grao)
>par(mfrow=c(2,2))
>plot(tabela)
```

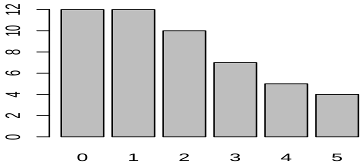
Another graph also with bars, but that can be used when having a non-numeric vector is the barplot(). Let's draw it for the same data

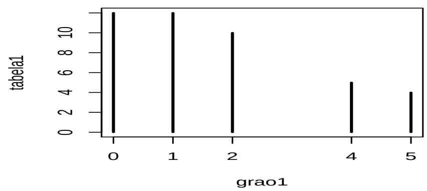
```
>barplot(tabela)  #Note a diferença no gráfico
>grao1<-c(1,2,0,0,1,4,2,5,1,1,5,0,2,2,2,1,0,0,
  2,2,5,5,0,1,0,0,1,1,2,0,4,1,4,0,4,2,1,1,0,2,0,4,1)
>tabela1<-table(grao1)
>plot(tabela1)
>barplot(tabela1)
```

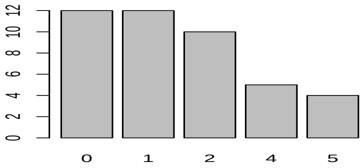
Graphs in R

Graphs with `plot()` e `barplot()`





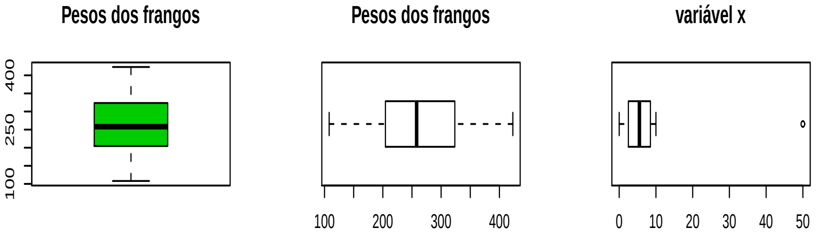




Graphs in R–o `boxplot()`

The `boxplot()`

```
> par(mfrow=c(1,3))
> boxplot(weight, main="Pesos dos frangos",col=3)
> boxplot(weight, main="Pesos dos frangos",horizontal=T)
> x <- c(0:10, 50)
> boxplot(x,horizontal=TRUE,main="variável x") # see an outlie
```

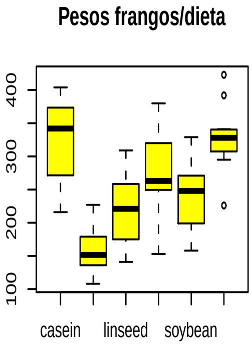


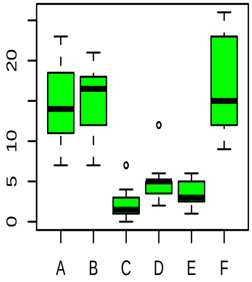
Graphs in R

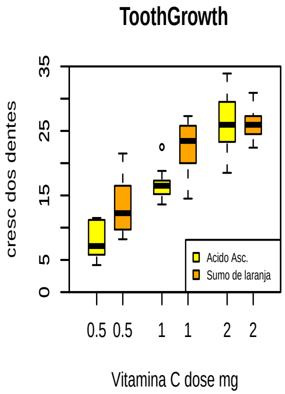
Side-by-Side boxplots

```
>par(mfrow=c(1,3))
>boxplot(weight~feed, main="Pesos frangos/dieta",col = "yellow")
>boxplot(count~spray,  col = "green",data=InsectSprays)
>boxplot(len ~ dose, data = ToothGrowth,
+       boxwex = 0.25, at = 1:3 - 0.2,
+       subset = supp == "VC", col = "yellow",
+       main = "ToothGrowth",
+       xlab = "Vitamina C dose mg",
+       ylab = "cresc dos dentes",
+       xlim = c(0.5, 3.5), ylim = c(0, 35), yaxs = "i")
>boxplot(len ~ dose, data = ToothGrowth, add = TRUE,
+       boxwex = 0.25, at = 1:3 + 0.2,
+       subset = supp == "OJ", col = "orange")
>legend("bottomright", c("Acido Asc.", "Sumo de laranja"),
+      fill = c("yellow", "orange"),cex=0.6)
```

Graphs in R


Pesos frangos/dieta
100
200
300
400
casein
linseed
soybean


0
5
10
20
A
B
C
D
E
F


ToothGrowth
cresc dos dentes
0
5
15
25
35
0.5
0.5
1
1
2
2
Vitamina C dose mg
Acido Asc.
Sumo de laranja

Exploratory analysis of bivariate data

Let us consider the `data frame`

```
>data(cars)
>head(cars)
```

Covariance and correlation coefficient

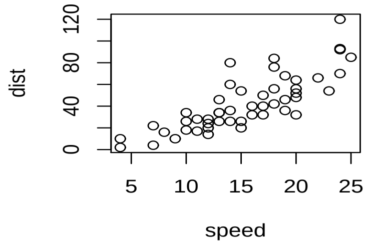
```
>attach(cars)
>cov(speed,dist)
>cor(speed,dist) # Pearson correlation coefficient.
```

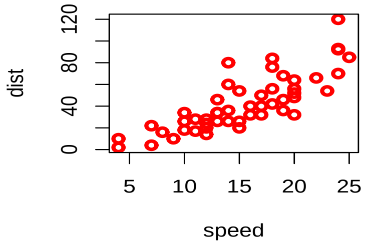
```
[1] 109.9469
[1] 0.8068949
```

Graphs for bivariate data

If (x_i, y_i) is a bivariate sample, the command `plot(x,y)` or `plot(y ~ x)` draw a scatter plot of y vs x.

```
>par(mfrow=c(1,2))
>plot(dist ~ speed, data=cars)
   # ou plot(cars$speed,cars$dist)
>plot(dist ~ speed, data = cars,col="red",lwd=3)
```





Graphs in R

Low-level graphical functions

- functions `points(x,y)` and `lines(x,y)` allow to add, respectively, points and points linked by lines;
- the function `abline(a,b)` adds a line with slope b and value y, for $x = 0$, a;
- the functions `abline(v=x)` e `abline(h=y)` allow to add vertical (in x) and horizontal (from y), respectively;
- the function `legend (title)` allows to add a caption (title) to the chart.
- the function `text(x,y,"dist.vel")` allows to write text in the coordinate `(x,y)`.

The simple linear regression model

In R the simple linear regression model uses the functions:
`lm(y∼x)` ou `lm(y∼1+x)`

```
> cars.lm <- lm(dist ~ speed)
> coef(cars.lm)

 (Intercept)       speed
 -17.579095    3.932409

> fitted(cars.lm)[1:5]
> predict(cars.lm, newdata = data.frame(speed = c(6, 8, 21)))

        1         2         3         4         5
-1.849460 -1.849460  9.947766  9.947766 13.880175

        1         2         3
 6.015358 13.880175 65.001489
```

The simple linear regression model

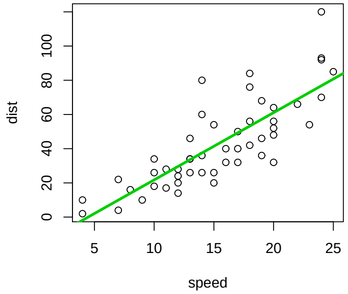
```
> summary(cars.lm)

Call:
lm(formula = dist ~ speed)
Residuals:
    Min      1Q  Median      3Q     Max
-29.069  -9.525  -2.272   9.215  43.201
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -17.5791     6.7584  -2.601   0.0123 *
speed         3.9324     0.4155   9.464 1.49e-12 ***

Residual standard error: 15.38 on 48 degrees of freedom
Multiple R-squared: 0.6511,     Adjusted R-squared: 0.6438
F-statistic: 89.57 on 1 and 48 DF,  p-value: 1.490e-12
```

Graphs in R

```
>plot(speed,dist)
>abline(cars.lm,col=3,lwd=3)
```


dist
speed