# Laboratory 1 - R and RStudio

R is a statistical computer program for performing statistical analysis introduced in 1996; it is an open source software (unlike Minitab, SPSS etc) that is extensively used for academic purposes.

In R you can program from the prompt line or using a GUI; we will use a graphical user interface: RStudio an open-source software too (used on Linux, Windows or Mac).

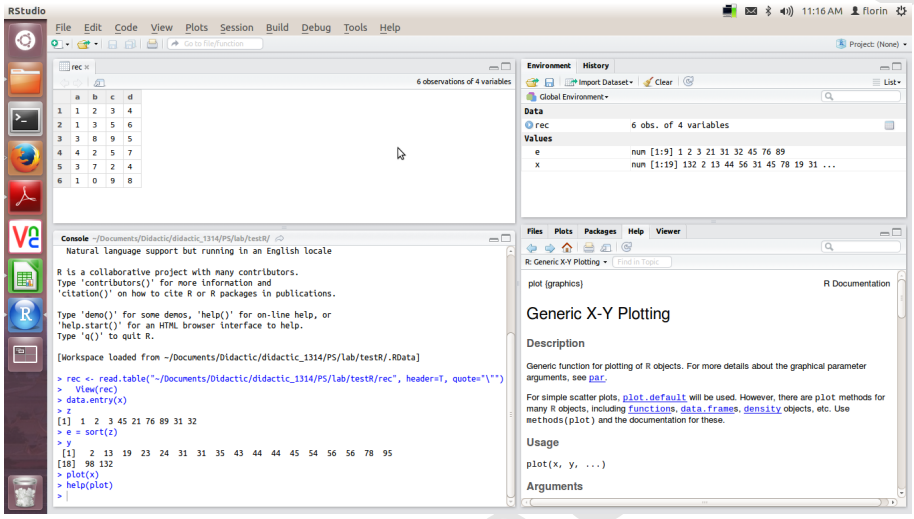


Figure 1: RStudio screenshot

RStudio has (see figure from above) four panels (starting from up left corner clockwise):

- an editing panel in which you can edit and run R scripts (which contain functions and commands) or data files;
- a panel that contains the history and can present the variables;
- a panel containing the help and in which the graphics are displayed;
- one panel that contains the prompt line (here you can execute R commands).

## 0.1 RStudio session

A session starts with setting of the working directory: **Session → Set Working Directory → Choose Directory** and will end by saving the workspace (from the dialog window "Save workspace image to /.RData?" choose "Save" in **Session → Save Workspace As**).

## 0.2 Variables and types

In R variables are vectors or matrices. Any variable can be displayed by calling its name. Types are: numerical, character strings (like "a43fdt") and boolean (TRUE or T, and FALSE or F).

**Assignment** There are two symbols for assignment in R: $=$ and $<-$ (without any spaces, it is recommend for compatibility with older versions of R).

**Create a vector** We present three different methods for creating a vector:

- by concatenation using function $c()$;
- as a sequence of consecutive integer numbers;

- or as a sequence in arithmetic progression for which we indicate the first number, the final one, and the length using function *seq*().

```
> x = c(1, 3, 2, 15, 6, 21, 34, 54, 7)
> x = c(T, T, F, T, F)
> x
[1] TRUE TRUE FALSE TRUE FALSE
> x = -5:13
> x
[1] -5 -4 -3 -2 -1 0 1 2 3 4 5 6 7 8 9
[16] 10 11 12 13
> x = seq(-3, 3, length=100)
```

You can access the elements of a vector like follows

```
> x = c(23, 21, 32, 25, 34, 19, 32, 45, 67)
> x[4] # the 4th element
[1] 25
x[2:6] # elements from 2 to 6, including 6
[1] 21 32 25 34 19
x[-3] # all elements except the 3rd
[1] 23 21 25 34 19 32 45 67
```

A vector can be edited using function *data.entry*(*vector*).

### 0.3 Arithmetic operations and predefined functions

We can perform such operations from the prompt line using variables or constants

```
> sin(1)
[1] 0.841471
> log(2)
[1] 0.6931472
> x = 3
> x^2
[1] 9
> exp(x)
[1] 20.08554
```

Vector operations are done componentwise

```
> x = c(1, 3, 2, 15, 6, 21, 34, 54, 7)
> y = c(22, 11, 32, 25, 54, 13, 27, 36, 2)
> x^2
[1] 1 9 4 225 36 441 1156 2916 49
> x + y
[1] [1] 23 14 34 40 60 34 61 90 9
```

R has mathematical and statistical functions for manipulating vectors, matrices, or simple variables.

```
> x < − c(1, 3, 2, 15, 6, 21, 34, 54, 7)
> length(x)
> [1] 9
> sort(x)
> [1] 1 2 3 6 7 15 21 34 54
> sqrt(x)
[1] 1.000000 1.732051 1.414214 3.872983 2.449490
[6] 4.582576 5.830952 7.348469 2.645751
> exp(x)
[1] 2.718282e+00 2.008554e+01 7.389056e+00
[4] 3.269017e+06 4.034288e+02 1.318816e+09
[7] 5.834617e+14 2.830753e+23 1.096633e+03
```

Information about a certain function can be obtained using $help(function\_name)$ on the prompt line.

## 0.4 Graphics

A simple two dimensional graphic can be obtained like follows: for example, if we want to represent the digital logarithm using as arguments 200 values from 0.001 to 10:

```
> x = seq(0.001, 10, length = 200)
> y = log2(x)
> plot(x, y, type = 'l', main='graphic', sub = 'subtitle', xlab ='x axis', ylab = 'y axis')
# graphic type: line
```

Another type uses bars of height equal with the given values. Let us represent the mass probability function for a binomial distribution $B(20, 0.4)$.

```
> n = 20  > x = seq(0,n,1) # x contains all integers from 0 to 20
> y = dbinom(x, n, 0.4)
> barplot(y, space = 0, main='barplot', sub = 'subtitle', xlab ='x axis', ylab = 'y axis')
# no space between bars
```

## 0.5 User defined functions

A function could be defined in the command line: suppose that we want to compute the variance of a distribution

```
> variance = function (x, p) {
+ expectation = sum(p*x);
+ variance = sum(p*(x - expectation)^2);
+ return (variance)
+ }
> y = c(23, 32, 31, 27, 27, 33, 25, 21)
> q = c(1/8, 1/16, 1/8, 1/16, 1/8, 1/16, 1/8, 5/16)
> variance(y, q)
```

It's more convenient to write such a function in an R script: **File** → **New File** → **R Script** , and than in the edit panel we write the code

```
variance = function(x, p) {
   expectation = sum(p*x);
   variance = sum(p*(x - expectation)^2);
   return (variance)
}
```

**RStudio.** After editing the script will be saved (**Ctrl+S**) cu with a name like "my_script.R" and can be loaded with **Code** → **Source File** (**Ctrl+Shift+O**) or from the command line with **source(script_file)**.

The same script can contain beside the definition of the function. a call to this function using different aruments; for example we can add to our script

```
> y = c(23, 32, 31, 27, 27, 33, 25, 21)
> q = c(1/8, 1/16, 1/8, 1/16, 1/8, 1/16, 1/8, 5/16)
> variance(y, q)
```

**RStudio.** Once loaded the script, a function defined in it can be executed from the command line: **variance(y, q)** or from the edit panel like this: we select the required lines and we execute them with **Ctrl+Enter**; the entire script can be executed with **Ctrl+Alt+R**.

A function can be modified in the edit panel or from the command line using $fix(function\_name)$

```
> fix(dispersie)
```

## 0.6 Manipulating data files

Suppose that a file "my_file" (which is in the current working directory, otherwise we must add the relative path to this file) contains an array of data (without any header); we can read the file and transform it in a vector.

```
> x = scan("my_file")
```

If the file contains a header (let us suppose that two columns have names "col1" and "col2"), then we execute

```
> y = read.table("my_file", header = T) # this object contains a header
> x1 = y['col1'] # this vector contains only the data from coloumn "col1"
> x2 = y['col2'] # this vector contains only the data from coloumn "col2"
```

We can read also csv (comma separated values) files:

```
> x = read.csv(file="date.csv", header = T)
```

## 0.7 Iterative and control structures

R has standard structures for iterations and control:

```
> if (condition){
+    statement
> }else
+    {
+    alternative
+    }
```

```
> for (var in sequence){
+    statement
> }
```

```
> while (condition){
+    statement
> }
```

The following function uses such structures:

```
vector_sqrt = function(x) {
  for(i in 1:length(x)) {
    if(x[i] > 0)
      x[i] = sqrt(x[i])
    else)
      x[i] = sqrt(-x[i])
  }
}
```

**Exercises.**

**1.** Introduce the following data (that represent the phone bills during a year in $) into a vector called $b$:

42 36 39 37 46 30 45 32 34 35 30 48

You can introduce the data from keyboard with

```
> b =scan(n = 7)
```

Find the maximum, the minum, the average, the sum of all bills, the ratio between the minimum and the maximum bills, the number of months having a bill of at least 40$, and the proportion of months having a bill under 40$.

**2.** Write four functions that, for a given vector $x$ (of length $n$), returns the following vectors

**(a)** $\dfrac{x_k}{\sum_{i=1}^{n} x_i}$, for $k = \overline{1,n}$;

**(b)** $\dfrac{x_k - \min\limits_{1\leqslant i\leqslant n} x_i}{\max\limits_{1\leqslant i\leqslant n} x_i}$, for $k = \overline{1,n}$;

**(c)** $\dfrac{\sum_{i=1}^{k} x_i}{\sum_{i=k+1}^{n} x_i}$, for $k = 1 \leqslant i \leqslant n-1$;

**(d)** $\dfrac{\max\limits_{1\leqslant i\leqslant k} x_i}{\min\limits_{k+1\leqslant i\leqslant n} x_i}$, for $k = 1 \leqslant i \leqslant n-1$.

(A new vector can be created with *vector*(). $n$ is the size of $x$.)

**3.** Create in the working directory a file "vector.txt" that a contains a column vector and modify the above functions in such a way that they read their vector from the file (These new functions will have as parameter: the name of the file.)

**4.** Write a function that prints and plots the mass probability (density) function of $B(n,p)$ and call this function for $(18, 0.25)$, $(40, 0.5)$ şi $(30, 0.8)$. (The function will have two parameters: $n$ and $p$.)

Hint: Use *barplot* and *dbinom* functions.

**5.** **(a)** Write a function that finds the maximum probability from the mass probability (density) function of $B(n,p)$. (The function will have two parameters: $n$ and $p$.)

**(b)** Write a function that compute the sum of the first $k$ probabilities from the mass probability (density) function of $B(n,p)$. (The function will have three parameters: $n, p$, and $k$, $1 \leqslant k \leqslant n$.)

(c) Write a function that compute $P(k \leqslant X \leqslant m)$, where $X : B(n,p)$ and $k, m$ are given natural numbers. (The function will have four parameters: $n, p, k$, and $m$, $0 \leqslant k \leqslant m \leqslant n$.)

Hint: Use *dbinom* function.

**6.** **(a)** Write a function which has to compute the sum of the first $n$ probabilities from the mass probability (density) function of $Geometric(p)$. (The function will have two parameters: $p$ and $n \geqslant 1$.)

**(b)** Write a function that compute $P(X \geqslant m)$, where $X : Geometric(p)$ and $m$ is a given natural number. (The function will have two parameters: $p$ and $m \geqslant 0$.)

Hint: Use *dgeom* function.

**7.** Write a function which has to print and plot the first $n$ probabilities from the mass probability (density) function of $Geometric(p)$. (The function will have two parameters: $p$ and $n \geqslant 1$.)

Hint: Use *barplot* and *dgeom* functions.

**8.** **(a)** Write a function which has to compute the sum of the first $n$ probabilities from the mass probability (density) function of $Poisson(\lambda)$. (The function will have two parameters: $\lambda$ and $n \geqslant 1$.)

**(b)** Write a function that finds the maximum probability among the first $n$ probabilities from the mass probability (density) function of $Poisson(\lambda)$. (The function will have two parameters: $\lambda$ and $n \geqslant 1$.)

**(c)** Write a function that compute $P(X \leqslant m)$, where $X : Poisson(\lambda)$ and $m$ is a given natural number. (The function will have two parameters: $\lambda$ and $m \geqslant 0$.)

Hint: Use *dpois* function.

**9.** Write a function that prints and plots the first $n$ probabilities from the mass probability (density) function of $Poisson(\lambda)$. (The function will have two parameters: $\lambda$ and $n \geqslant 1$.)

Hint: Use *barplot* and *dpois* functions.

**10.** Write functions that read the two columns (called "AA" and "BB") of the file "test.txt", transform them in two vectors $x$ and $y$ and, then

**(a)** plot the pairs $(x_i, y_i)$, using *plot*() (try also some of the parameters of the function *plot*());

**(b)** compute the vector having the elements $(x_k * y_k)$, $k = \overline{1,n}$.

**(c)** compute and print the vector having the elements $\dfrac{|x_k - y_k|}{\sum_{i=1}^{n}(x_i - y_i)^2}$, $k = \overline{1,n}$.

**(d)** computes and prints the closest point $(x_i, y_i)$ to a given point $(p, q)$.

Hint: Use *plot* and *help* functions.