R software installation and first steps

In this module you will find the step by step instruction to install the R software and receive a basic imprinting of R language.

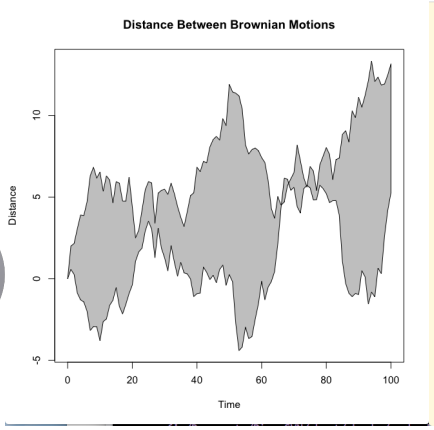
What is R?

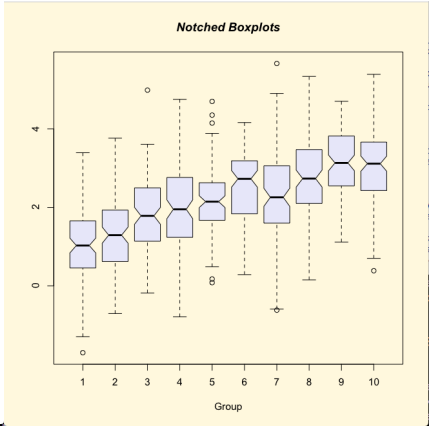
R is a programming language and software environment for statistical computing and graphics. The R language is widely used among statisticians and data miners for developing statistical software and data analysis. Polls, surveys of data miners, and studies of scholarly literature databases show that R's popularity has increased substantially in recent years.[Read more from Wikipedia...]

R main features are:

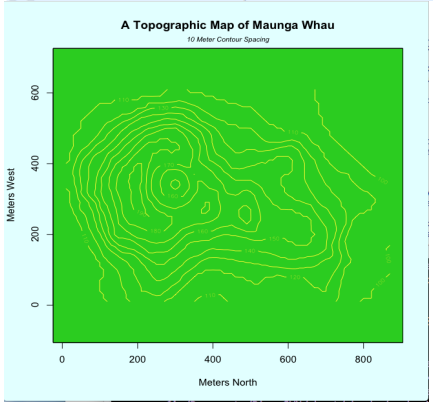
- **object-oriented** programming,
- provides **calculations on matrices**,
- excellent **graphics** capabilities,
- supported by a **large user network**.

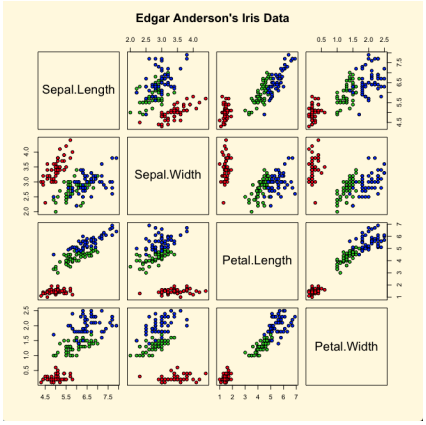
In this first module, you are going to find information and some help to **install** the programme, and you are going to be introduced to **R language**.





```
> summary(detg.mod, correlation = TRUE, symbolic.cor = TRUE)

Call:
glm(formula = terms(Fr ~ M.user * Temp * Soft + Brand * M.user *
    Temp, keep.order = TRUE), family = poisson, data = detg)

Deviance Residuals:
     Min        1Q    Median        3Q       Max
-0.91365  -0.35585   0.00253   0.33027   0.92146

Coefficients:
                           Estimate Std. Error z value Pr(>|z|)
(Intercept)                 4.14887    0.10603  39.128  < 2e-16 ***
M.userY                    -0.40521    0.16188  -2.503  0.01231 *
TempHigh                   -0.44275    0.17121  -2.586  0.00971 **
M.userY:TempHigh           -0.12692    0.26257  -0.483  0.62883
SoftMedium                  0.05311    0.13308   0.399  0.68984
SoftSoft                    0.05311    0.13308   0.399  0.68984
M.userY:SoftMedium          0.08323    0.19685   0.423  0.67245
M.userY:SoftSoft            0.12169    0.19591   0.621  0.53449
TempHigh:SoftMedium        -0.30442    0.22239  -1.369  0.17104
TempHigh:SoftSoft          -0.30442    0.22239  -1.369  0.17104
M.userY:TempHigh:SoftMedium  0.21189    0.31577   0.671  0.50220
M.userY:TempHigh:SoftSoft   -0.20387    0.32540  -0.627  0.53098
BrandM                      -0.30647    0.10942  -2.801  0.00510 **
M.userY:BrandM               0.40757    0.15961   2.554  0.01066 *
TempHigh:BrandM              0.04411    0.18463   0.239  0.81119
M.userY:TempHigh:BrandM      0.44427    0.26673   1.666  0.09579 .
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for poisson family taken to be 1)

    Null deviance: 118.627  on 23  degrees of freedom
Residual deviance:   5.656  on  8  degrees of freedom
AIC: 170.07

Number of Fisher Scoring iterations: 4

Correlation of Coefficients:

(Intercept)                1
M.userY                    , 1
TempHigh                   , . 1
M.userY:TempHigh           . , , 1
SoftMedium                 , . .   1
SoftSoft                   , . .   . 1
M.userY:SoftMedium         . ,   . , . 1
M.userY:SoftSoft           . ,   . . , . 1
TempHigh:SoftMedium        .   , . . . .   1
TempHigh:SoftSoft          .   , . . .   . . 1
M.userY:TempHigh:SoftMedium   . . . .   , . , . 1
M.userY:TempHigh:SoftSoft     . . .   . . , . , . 1
BrandM                     .                       1
M.userY:BrandM               .                     , 1
TempHigh:BrandM                . .                 . . 1
M.userY:TempHigh:BrandM        . .                 . . , 1
attr(,"legend")
[1] 0 ‘ ’ 0.3 ‘.’ 0.6 ‘,’ 0.8 ‘+’ 0.9 ‘*’ 0.95 ‘B’ 1
```




Edgar Anderson's Iris Data
Sepal.Length
Sepal.Width
Petal.Length
Petal.Width

Installing R on Windows/1

This course is centered on the installation of the software on **Windows system**. However, the software can be handy installed in every platform, with few differences in the management of the packages.

Notice that software version indicated in the images below has to be intended as an example since new version are frequently released.

1. **Access the R Project site:**

   https://cran.r-project.org

2. **Choose the proper version for your operating system. Here, the link "Download R for Windows" is chosen.**

CRAN
Mirrors
What's new?
Task Views
Search

About R
R Homepage
The R Journal

Software
R Sources
R Binaries
Packages
Other

Documentation
Manuals
FAQs
Contributed

The Comprehensive R Archive Network

Download and Install R

Precompiled binary distributions of the base system and contributed packages, **Windows and Mac** users most likely want one of these versions of R:

- Download R for Linux
- Download R for (Mac) OS X
- Download R for Windows

R is part of many Linux distributions, you should check with your Linux package management system in addition to the link above.

Source Code for all Platforms

Windows and Mac users most likely want to download the precompiled binaries listed in the upper box, not the source code. The sources have to be compiled before you can use them. If you do not know what this means, you probably do not want to do it!

- The latest release (Thursday 2017-09-28, Short Summer) R-3.4.2.tar.gz, read what's new in the latest version.
- Sources of R alpha and beta releases (daily snapshots, created only in time periods before a planned release).
- Daily snapshots of current patched and development versions are available here. Please read about new features and bug fixes before filing corresponding feature requests or bug reports.
- Source code of older versions of R is available here.
- Contributed extension packages

Questions About R

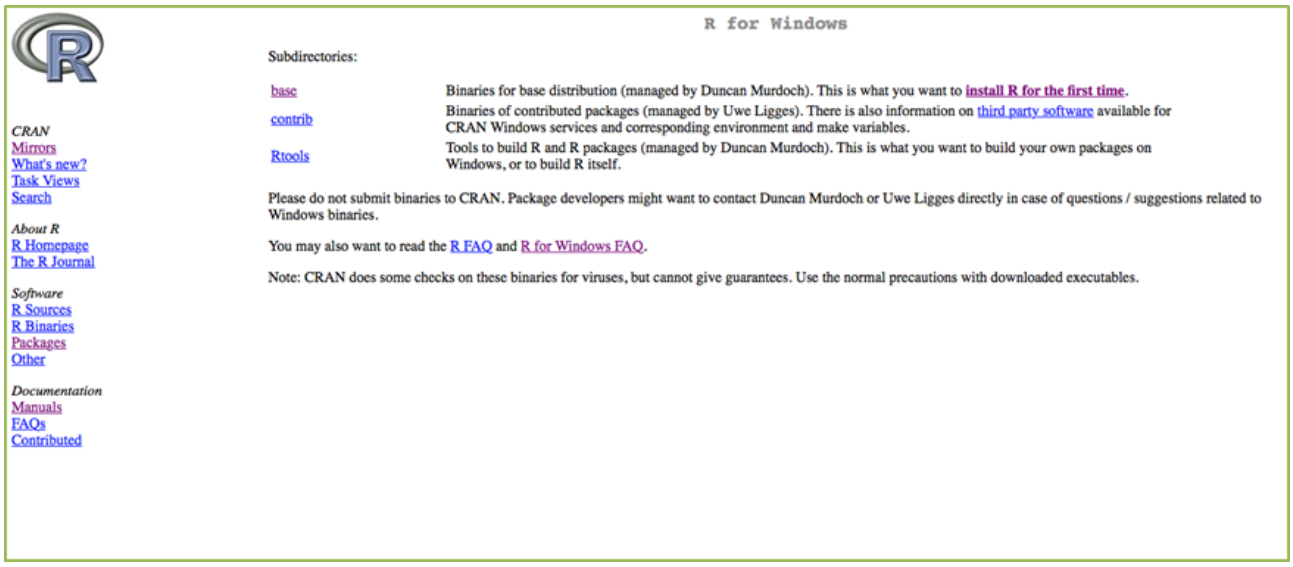
- If you have questions about R like how to download and install the software, or what the license terms are, please read our answers to frequently asked questions before you send an email.

Installing R on Windows/2

1. ~~Access the R Project site:~~ https://cran.r-project.org~~;~~

2. ~~Choose proper the version for your operating system. Here the link "Download R for Windows" is chosen;~~

3. **Once opened the new page, choose the link "base" that will direct you to another page containing the last version of R.**



Installing R on Windows/3

1. ~~Access the R Project site:~~ https://cran.r-project.org
2. ~~Choose the proper version for your operating system. Here the link "Download R for Windows" is chosen.~~
3. ~~Once opened the new page, choose the link "base" that will direct you to another page with containing the last version of R;~~
4. **Download the file and start the installation process. At the end of the installation, an R icon is added to your desktop.**

CRAN
Mirrors
What's new?
Task Views
Search

About R
R Homepage
The R Journal

Software
R Sources
R Binaries
Packages
Other

Documentation
Manuals
FAQs
Contributed

R-3.4.2 for Windows (32/64 bit)

Download R 3.4.2 for Windows (75 megabytes, 32/64 bit)
Installation and other instructions
New features in this version

If you want to double-check that the package you have downloaded matches the package distributed by CRAN, you can compare the md5sum of the .exe to the fingerprint on the master server. You will need a version of md5sum for windows: both graphical and command line versions are available.

Frequently asked questions

- Does R run under my version of Windows?
- How do I update packages in my previous version of R?
- Should I run 32-bit or 64-bit R?

Please see the R FAQ for general information about R and the R Windows FAQ for Windows-specific information.

Other builds

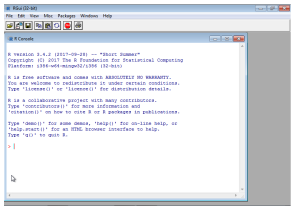
- Patches to this release are incorporated in the r-patched snapshot build.
- A build of the development version (which will eventually become the next major release of R) is available in the r-devel snapshot build.
- Previous releases

Note to webmasters: A stable link which will redirect to the current Windows binary release is <CRAN MIRROR>/bin/windows/base/release.htm.

Last change: 2017-09-28, by Duncan Murdoch

Installing R on Windows/4

1. ~~Access the R Project site:~~ https://cran.r-project.org
2. ~~Choose the version for your operating system. As said before, this course is on the installation on Windows system. So here the link "Download R for Windows" is chosen.~~
3. ~~Once opened the new page, choose the link "base" that will direct you to another page with containing the last version of R;~~
4. ~~Download the file and start the installation process. At the end of the installation, an R shape icon is added on your desktop;~~



5. **Double click on the icon on your desktop to open the R Console.**

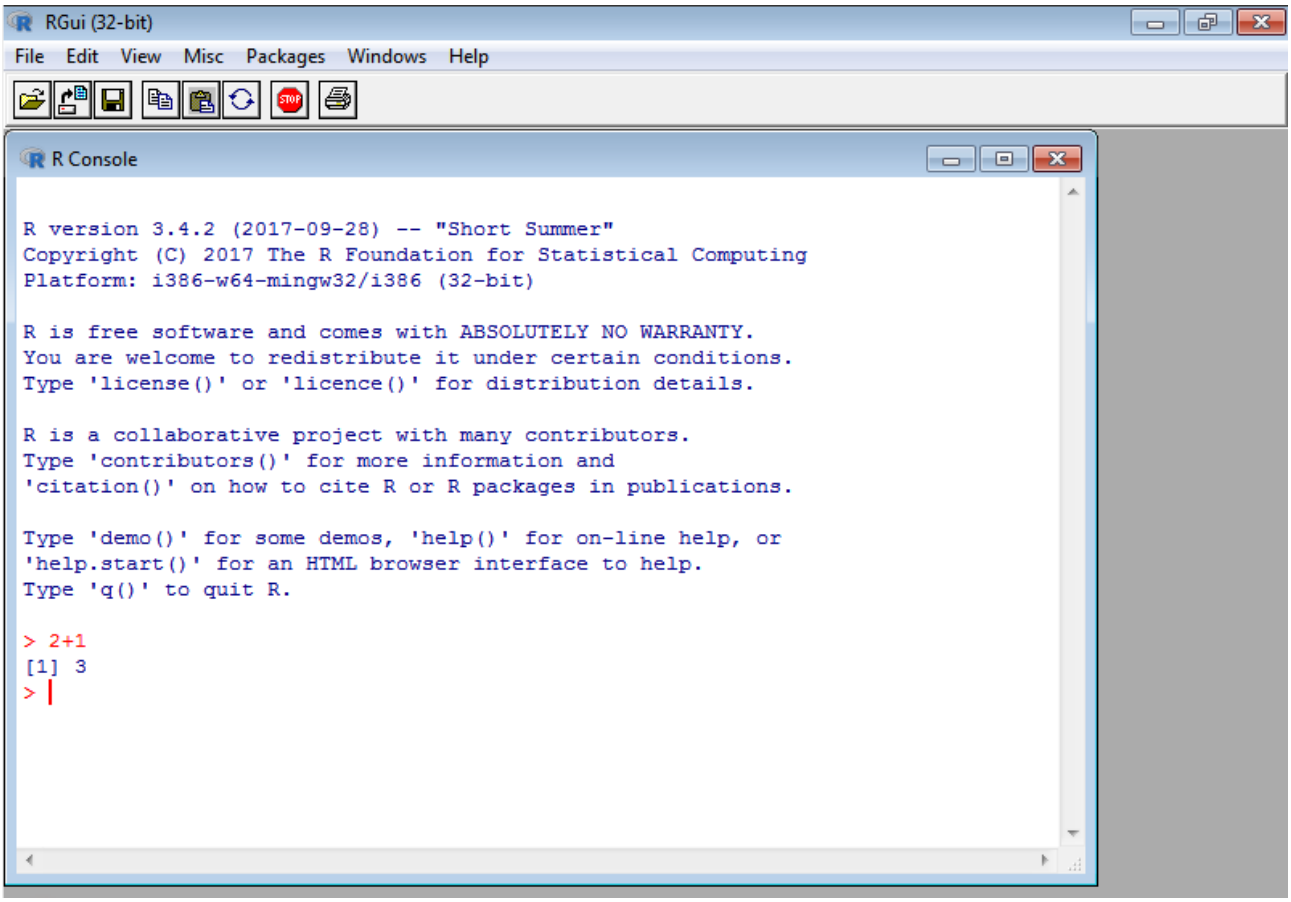
R Console

R console is the **command line** interface of the R engine.

In this mode, R can be used as a very simple **calculator**: for example type

1+2

and press the Return key.



It is quite simple till now, isn't it?

From now on, you are requested to keep R console opened on your PC and repeat lesson's examples side by side.

The Kernel of R language (in a nutshell)

Everything in R is an object.

Every
object in R is a member of a class (e.g. character vectors, numeric vectors, data frames, lists, arrays, functions, etc.)

Every object in R has a type (e.g. single, double, etc.).

- **Objects**:
    - **vector** [dim1] --> "dim" stands for "dimension"
    - **matrix** [dim1, dim2]
    - **array** [dim1, dim2, dim3]
    - **data frame** (a better organized matrix)
    - **list** (objects of different nature and length in a unique store)
    - **date** (special objects for treating dates)
    - **time series**
    - ....
- **Functions**:
    - utilities to **transform** objects --> e.g. **log(x)**, **sort(x)**
    - to **analyze** objects --> e.g. **summary(x)**, **boxplot(x)**
    - to **build a statistical model** --> e.g. **lm(y~x)**, **glm(y~x)**

DON'T WORRY IF IT IS NOT CLEAR AT THIS STAGE, PROCEED!

The kernel of R language: Arithmetic expressions

Arithmetic expressions

Chunk 1 (objects class and type of)

```
x = 12.5   # is equivalent of  x <- 12.5
x
```

```
## [1] 12.5
```

```
class(x)
```

```
## [1] "numeric"
```

```
typeof(x)
```

```
## [1] "double"
```

Chunk 2 (integer)

Integer vectors exist so that data can be passed to **C** or **Fortran** code which expects them:

```
k = 1
k
```

```
## [1] 1
```

```
is.integer(k)  #is.integer(x) does not test if x contains integer numbers!
```

```
## [1] FALSE
```

```
is.double(k)
```

```
## [1] TRUE
```

```
y = as.integer(k)
y
```

```
## [1] 1
```

```
as.integer(3.14)
```

```
## [1] 3
```

```
as.integer("5.27")
```

```
## [1] 5
```

```
as.integer("Donald")
```

```
## Warning: si è prodotto un NA per coercizione
```

```
## [1] NA
```

Exercise-basic-commands

Set **x=c(2,7).**

What does appear in the console if you run **is.numeric(x)**?

What if you run **class(x)**?

What if you run **str(x)** ?

The Kernel of R language: expressions

R code is composed of a series of **expressions** such as:

- **assignment statements**

```
x <- 4
paste("class of x is",class(x),"and type is",typeof(x))
```

```
## [1] "class of x is numeric and type is double"
```

- **arithmetic expressions**

```
10 %% 4
```

```
## [1] 2
```

- **conditional statements**

```
if (4 > 2) "yes" else "no"
```

```
## [1] "yes"
```

or

```
if (4 > 2) 4*2
```

```
## [1] 8
```

Exercise-basic-commands2

Set **x<-rep(0,times=10)**. Which object do we have build?

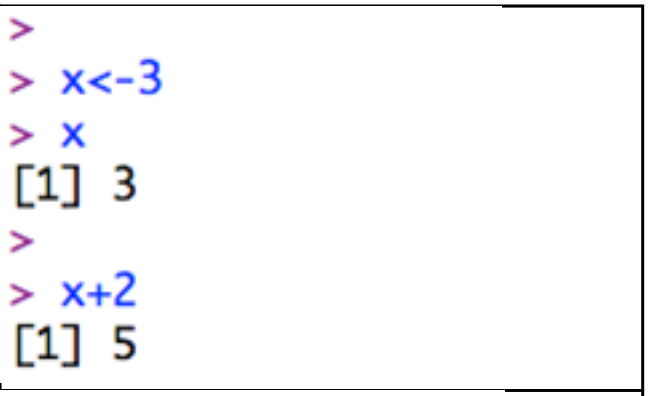
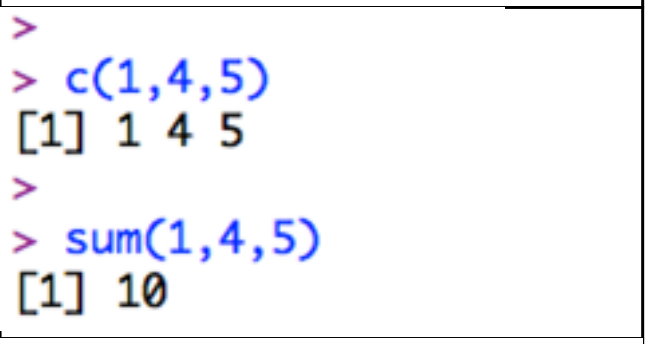
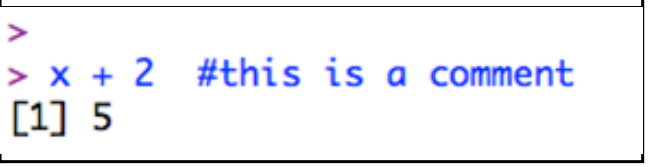
A vector of 10 numbers.

An object of class "numeric".

A NULL object.

The Kernel of R language: assignment

KEEP PRACTICING ON YOUR OWN R CONSOLE!

Object Assignment You can store values into an object with the assignment operator **<- or =** to access it later. **x**becomes a variable and can now be used in expressions in place of the original value.	
Functions You can call a function by typing its name. In the example, the function **c()** combines 3 numeric values into a vector. Let's try using the **sum()** function to add up a few numbers.	
Comments All text after the pound sign # within the same line is considered a comment.	

The kernel of R language: documentation

Help and documentation

There is a large amount of (free) documentation and help available. Some help is automatically installed.

Typing in the console:

>help(sum)

you will have help about the **sum()** function.

sum {base} R Documentation

Sum of Vector Elements

Description

`sum` returns the sum of all the values present in its arguments.

Usage

```
sum(..., na.rm = FALSE)
```

Arguments

`...` numeric or complex or logical vectors.

`na.rm` logical. Should missing values (including `NaN`) be removed?

Details

This is a generic function: methods can be defined for it directly or via the `Summary` group generic. For this to work properly, the arguments `...` should be unnamed, and dispatch is on the first argument.

If `na.rm` is `FALSE` an `NA` or `NaN` value in any of the arguments will cause a value of `NA` or `NaN` to be returned, otherwise `NA` and `NaN` values are ignored.

Typing:

>example(sum)

you will find some examples of how the function can be used.

Notice how this function works when NA are present in the data.

```
sum> ## Pass a vector to sum, and it will add the elements together.
sum> sum(1:5)
[1] 15

sum> ## Pass several numbers to sum, and it also adds the elements.
sum> sum(1, 2, 3, 4, 5)
[1] 15

sum> ## In fact, you can pass vectors into several arguments, and everything gets added.
sum> sum(1:2, 3:5)
[1] 15

sum> ## If there are missing values, the sum is unknown, i.e., also missing, ....
sum> sum(1:5, NA)
[1] NA

sum> ## ... unless  we exclude missing values explicitly:
sum> sum(1:5, NA, na.rm = TRUE)
[1] 15
>
```

Typing:

> ??cumulative

you will obtain an interactive list

*of **packages::functions** that contain the*

*word **cumulative** in the description text.*

Search Results

Help pages:

base::cumsum	Cumulative Sums, Products, and Extremes
stats::cpgram	Plot Cumulative Periodogram
stats::ecdf	Empirical Cumulative Distribution Function

CONSIDER THE USE OF WEB AS A USEFUL OPTION TO FIND HELP, AS WELL!

The kernel of R language: manipulation

CREATE AND MANIPULATE THE OBJECTS SHOWN IN EXAMPLES IN YOUR CONSOLE.

Vector

A vector is simply a **collection of values**.
Vector's elements can be **numbers**, **strings**, **logical values** (TRUE, FALSE), or any other type, as long as they're **all the *same* type**.

```
>
> num<-c(2,5,7)
> string<-c("a","b","c")
>
> num
[1] 2 5 7
> string
[1] "a" "b" "c"
```

Vector indexing

You can retrieve an individual value within a vector by providing its **numeric index** in square brackets. Try getting the third value from vector named “string”.
You can pick multiple values from a vector. Try getting the first *and* third elements.

```
>
> string[3]
[1] "c"
>
> string[c(1,3)]
[1] "a" "c"
>
```

Vector names

You can name vector's elements by using **names()** function.

```
> num
[1] 2 5 7
>
> names(num)<-c("first", "second", "third")
> num
 first second  third
     2      5      7
>
```

The kernel of R language: Sequences

Chunk 4 (sequences)

You can also create a vector using sequences.

```
X.ahead = 1:10
X.ahead
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10
```

```
X.seq = seq(1,10,by=0.5)
X.seq
```

```
##  [1]  1.0  1.5  2.0  2.5  3.0  3.5  4.0  4.5  5.0  5.5  6.0  6.5  7.0  7.5
## [15]  8.0  8.5  9.0  9.5 10.0
```

```
X.seq2 = seq(1,10,length.out = 20)
X.seq2
```

```
##  [1]  1.000000  1.473684  1.947368  2.421053  2.894737  3.368421  3.842105
##  [8]  4.315789  4.789474  5.263158  5.736842  6.210526  6.684211  7.157895
## [15]  7.631579  8.105263  8.578947  9.052632  9.526316 10.000000
```

```
(X.rep = rep(NA,length = 10))    # round brackets for printing the object directly
```

```
##  [1] NA NA NA NA NA NA NA NA NA NA
```

```
(X.rep2 = rep(c("RTC2017","WMO"),each = 10))
```

```
##  [1] "RTC2017" "RTC2017" "RTC2017" "RTC2017" "RTC2017" "RTC2017" "RTC2017"
##  [8] "RTC2017" "RTC2017" "RTC2017" "WMO"     "WMO"     "WMO"     "WMO"
## [15] "WMO"     "WMO"     "WMO"     "WMO"     "WMO"     "WMO"
```

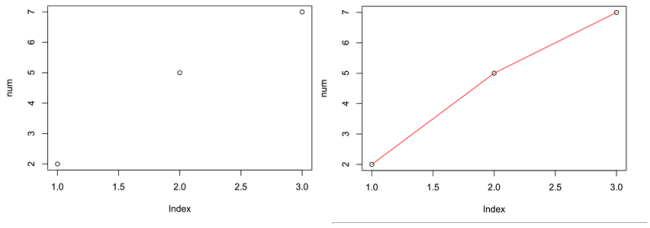
```
(X.rep3 = rep(paste("RTC2017","WMO"),each = 10))  # paste() instead of c()
```

```
##  [1] "RTC2017 WMO" "RTC2017 WMO" "RTC2017 WMO" "RTC2017 WMO" "RTC2017 WMO"
##  [6] "RTC2017 WMO" "RTC2017 WMO" "RTC2017 WMO" "RTC2017 WMO" "RTC2017 WMO"
```

The kernel of R language: visualizing

Plotting vectors

The **plot()** function draws a chart of vector's values.



>

> plot(num)

You can add a line joining points in the existing plot by using the **lines()** function.

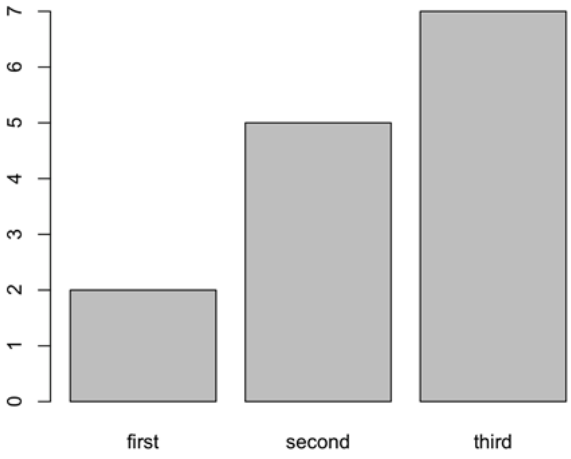
>

>lines(num, col="red")

The **barplot()** function draws a barchart of vector's values.

>

> barplot(num)



The kernel of R language: matrix

Matrix

A **matrix** is a collection of data elements arranged in a two-dimensional rectangular layout. The following is an example of a matrix with 2 rows and 3 columns.

I recommend you to have a look at in-line help to clarify what the three arguments of the function represent (TIP: type ?matrix into console).

```
> MAT<-matrix(c(1:6),2,3)
> MAT
     [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6
```

Matrix indexing

Getting values from matrices isn't that different from vectors; you just have to provide two indices instead of one. You can get an entire row of the matrix by omitting the column index (but keep the comma). Try retrieving the second row.

```
> MAT[2,3]
[1] 6
> MAT[2,]
[1] 2 4 6
```

The kernel of R language: data.frame

Data frame

A data frame is used for storing data tables. It is a **list of vectors of equal length**. For example, the following variable **df** is a data frame containing three vectors **n**, **s**, **b**.

```
> n<-c(2,3,5)
> s<-c("aa", "bb", "cc")
> b<-c(TRUE, FALSE, TRUE)
> df<-data.frame(n, s, b)       # df is a data frame
>
>
> df
  n  s     b
1 2 aa  TRUE
2 3 bb FALSE
3 5 cc  TRUE
>
```

Data frame indexing

You can get individual columns typing the data frame name, a dollar sign (or double square brackets), and the column name.

```
> df$s
[1] aa bb cc
Levels: aa bb cc
> df$n
[1] 2 3 5
```

```
> mean(df$n)
[1] 3.333333
> mean(df[["n"]])
[1] 3.333333
>
```

How to extract elements from a data frame object.

Does the indexing of data frame by using the column/row number works?

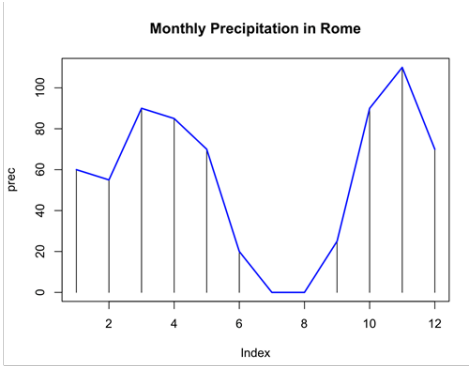
True

False

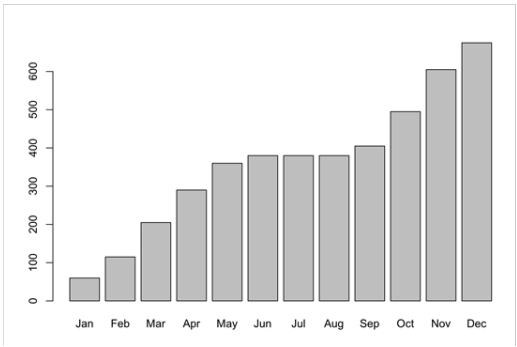
The kernel of R language…. and now YOU MOVE!

Think of the monthly distribution of precipitation at your place and populate an object named “prec” with the 12 values (just guess because we’re going to compare them with the observed data).

```
>
> # THIS IS MY GUESS FOR ROME
> prec<-c(60,55,90,85,70,20,0,0,25,90,110,70)
> names(prec)<-month.abb
> prec
Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
 60  55  90  85  70  20   0   0  25  90 110  70
> plot(prec, type="h", main="Monthly Precipitation in Rome")
> lines(prec, col="blue", lwd=2)
```




```
> preccum<-cumsum(prec)
> preccum
Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
 60 115 205 290 360 380 380 380 405 495 605 675
> barplot(preccum)
```



The kernel of R language: script file

Writing commands into a script file

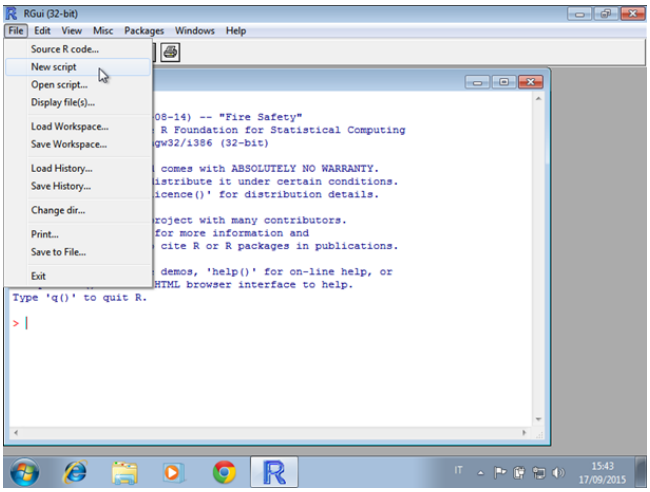
- Create and save a script file by means of R editor
- Copy and paste the following 4 lines in the created file:

prec<-c(60,55,90,85,70,20,0,0,25,90,110,70)

names(prec)<-month.abb

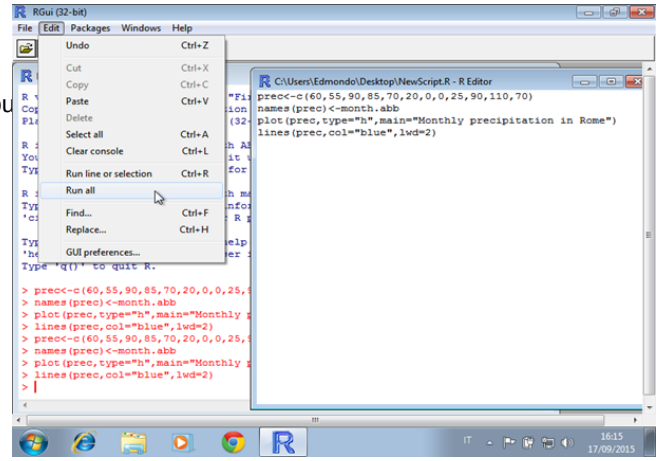
plot(prec, type="h", main="Monthly Precipitation in Rome")

lines(prec, col="blue", lwd=2)



Executing commands from a script file

- From "Edit" menu, select and click "Run all"……you should visualize a new window with the usual plot.



Exercise: knowing basic commands

The **history()** command.

When running the **history()** command in the R Console, you get:

When running the **tail(prec)** command in the R Console, you get:

When running the **head(prec)** command in the R Console, you get:

R packages

Packages are collections of **Rfunctions**, **data**, and **compiled code** in a well-defined format. The directory where packages are stored is called the **library**.

For instance, the function **mean()** is included in *base* package and **lm()** for linear trend model is in *stats* package.

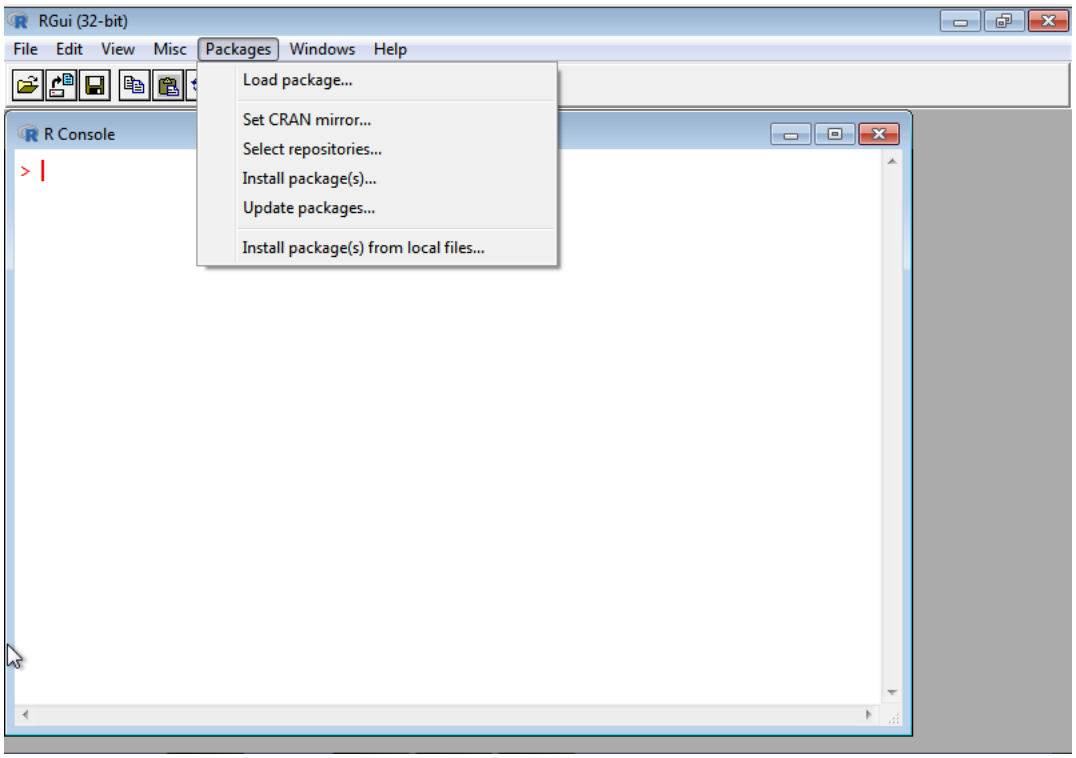
Both *base* and *stats* are preloaded in the R console since **R** comes with a **standard set of packages**. However, **many others are available for download and installation**.

Non standard packages have to be loaded into the session after installed in order to be used.

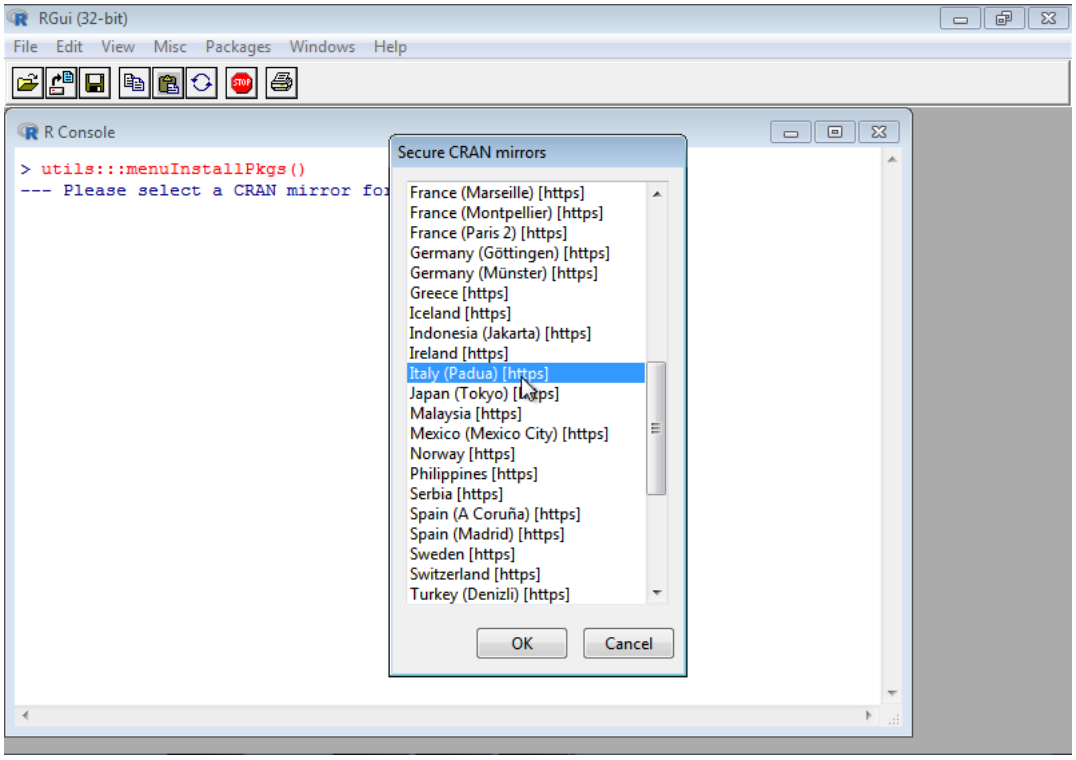
Installing Packages

On MS Windows:

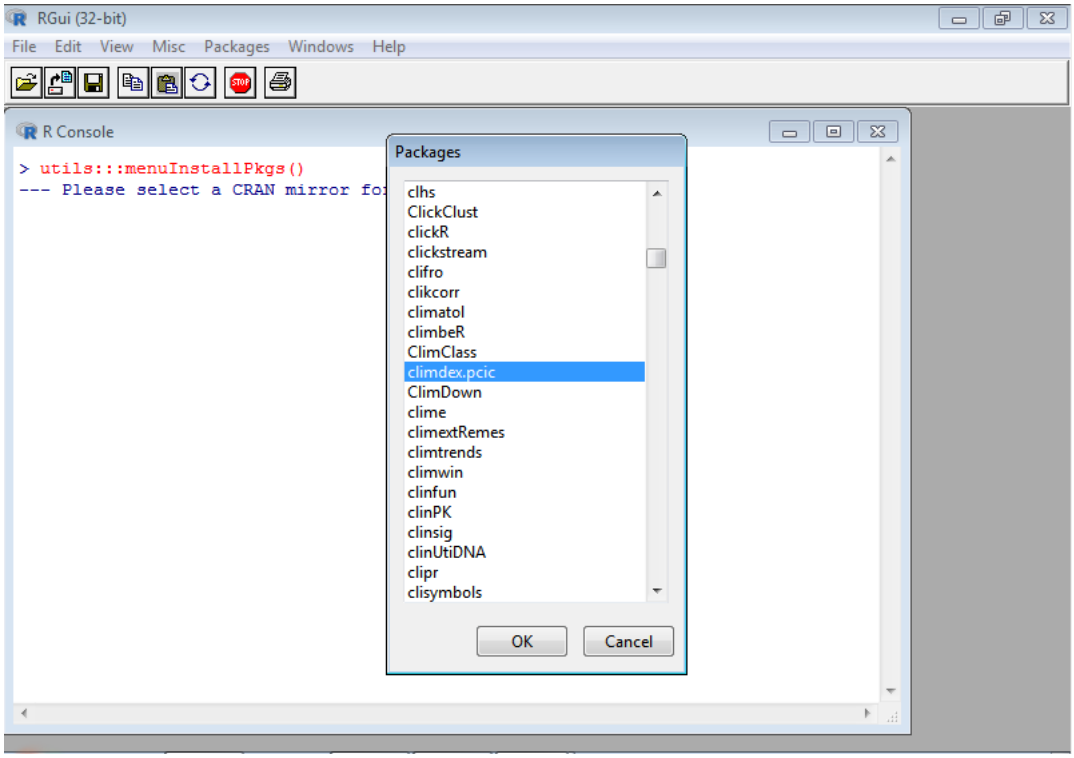
- Choose **Install Packages** from the **Packages** menu;



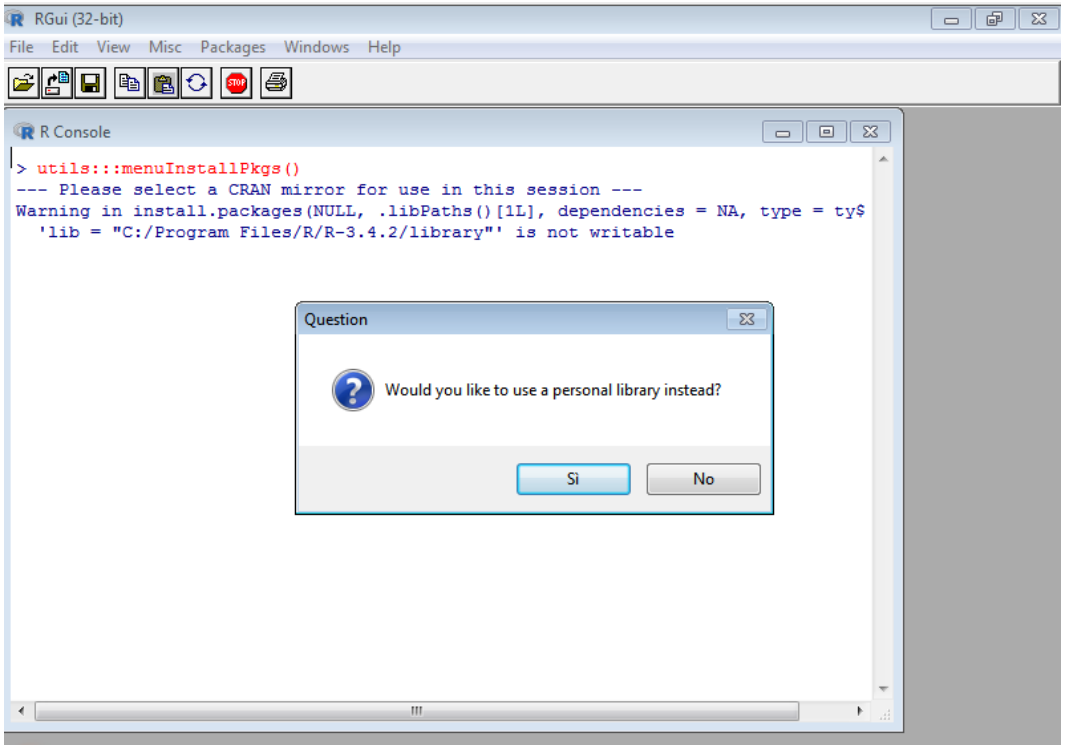
- Select a **CRAN Mirror**. (e.g. Italy);



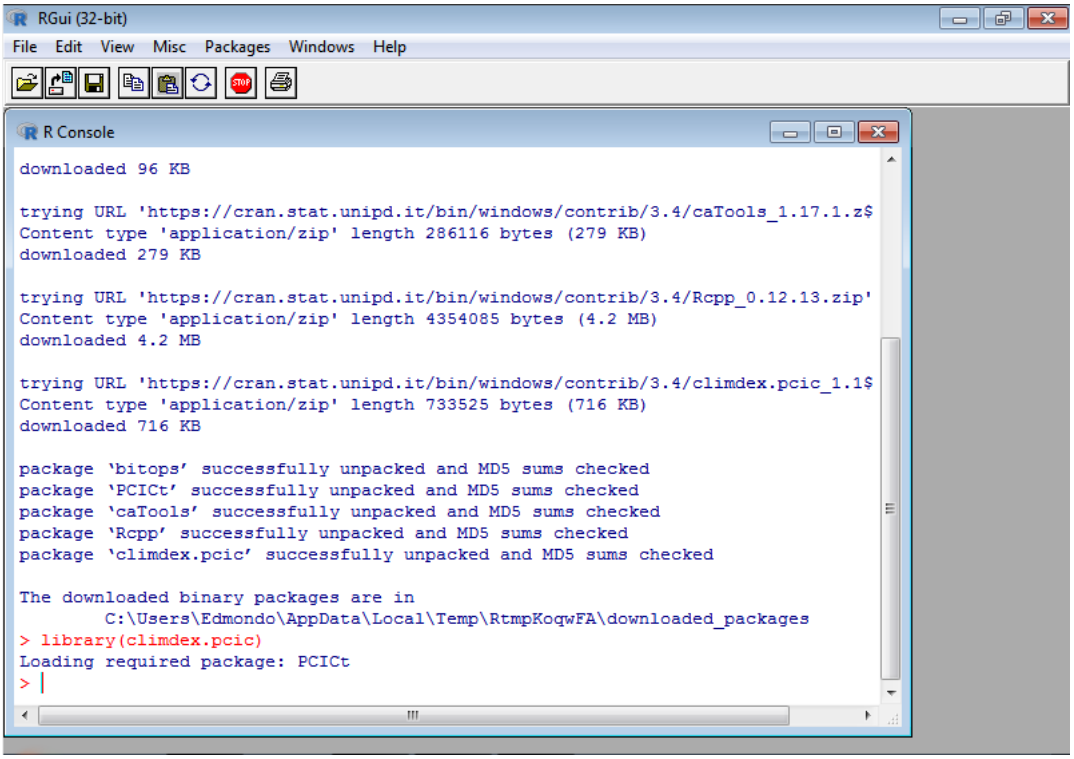
- Select a package. (e.g. climdex.pcic; https://cran.r-project.org/web/packages/climdex.pcic/)



- When prompted to the following pops-up, tick "Yes" so that a new directory will be created where this and other personal packages will be stored (it will be useful whenever you will decide to update R to newer version).



- Finally, type in the console **library("climdex.pcic")** to load it in the engine.



Summary

Is everything all right?

At this point you should be able to:

- install packages
- build and name vectors, matrix and data frames
- extract elements from objects
- draw a basic plot
- execute commands by launching a source file (.R)

In the next module, you will be introduced to use a more friendly editor (Rstudio), which integrates to the R engine and you will learn to work on time series.

The end

Welcome to the cruel world of

DATA ANALYSIS AND MANIPULATION

…to be continued