Chapter 1: Introduction and preliminaries

Contents

1 What is R?

R is a system for statistical computation and graphics. It consists of a language plus a run-time environment with graphics, a debugger, access to certain system functions, and the ability to run programs stored in script files.

R is easily extensible using a package library system. A wide-ranging and extensive set of contributed packages is also available from the R archive network (http://cran.r-project.org/).

2 Download R and RStudio

2.1 Installation of R software

Please be aware to have the most recent version installed!

Install R from http://cran.r-project.org.

2.2 Install RStudio

RStudio is a free, open source interface for working with **R**. You can download it from http://rstudio.com/, install, and run it on your computer.

It has 4 fields:

1. script space + data viewer
2. workspace browser + code history window
3. console: executed codes + output
4. files and folders from the work directory window + graphical window + window with the list of installed packages + help window (integrated with R).

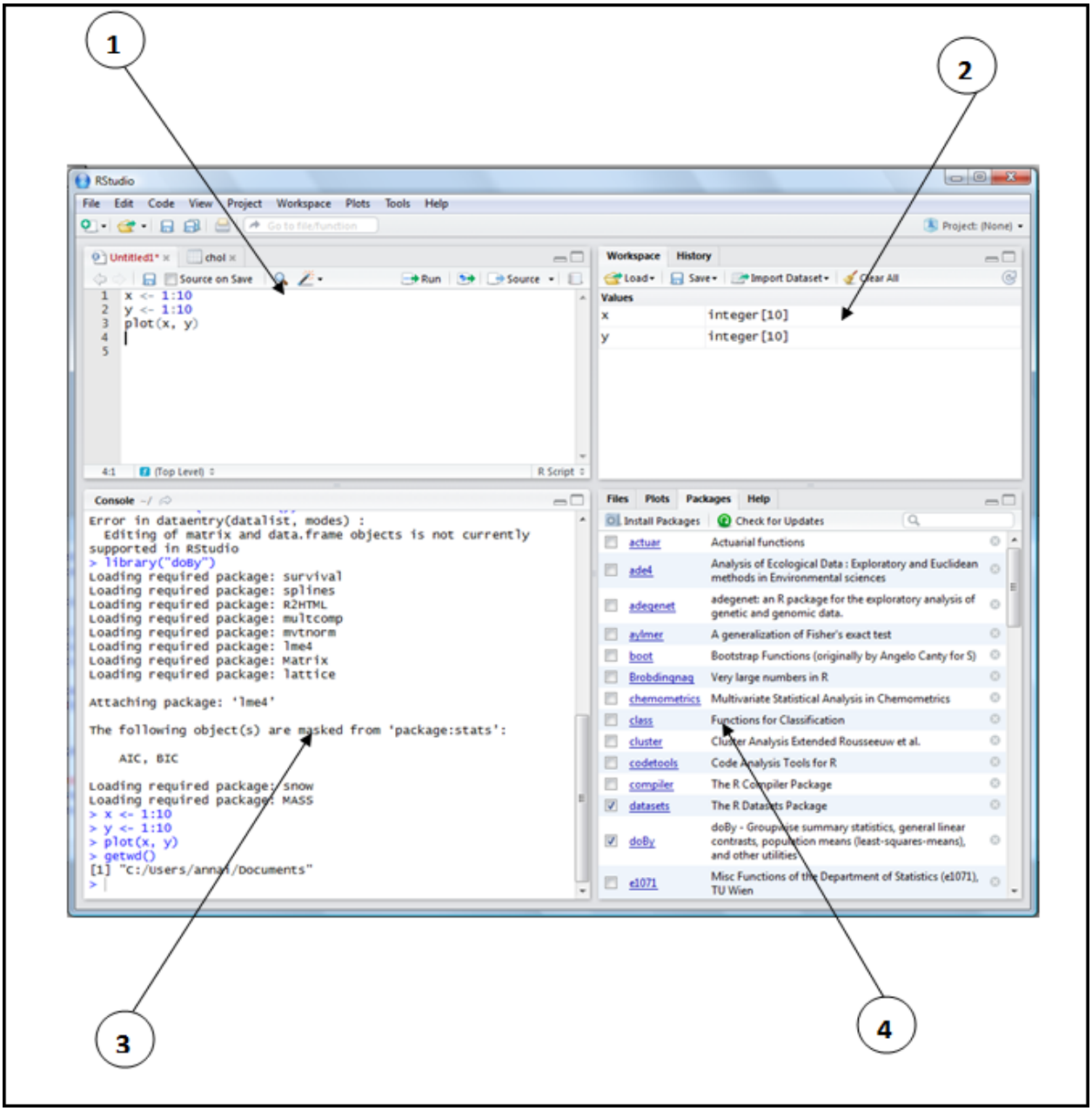
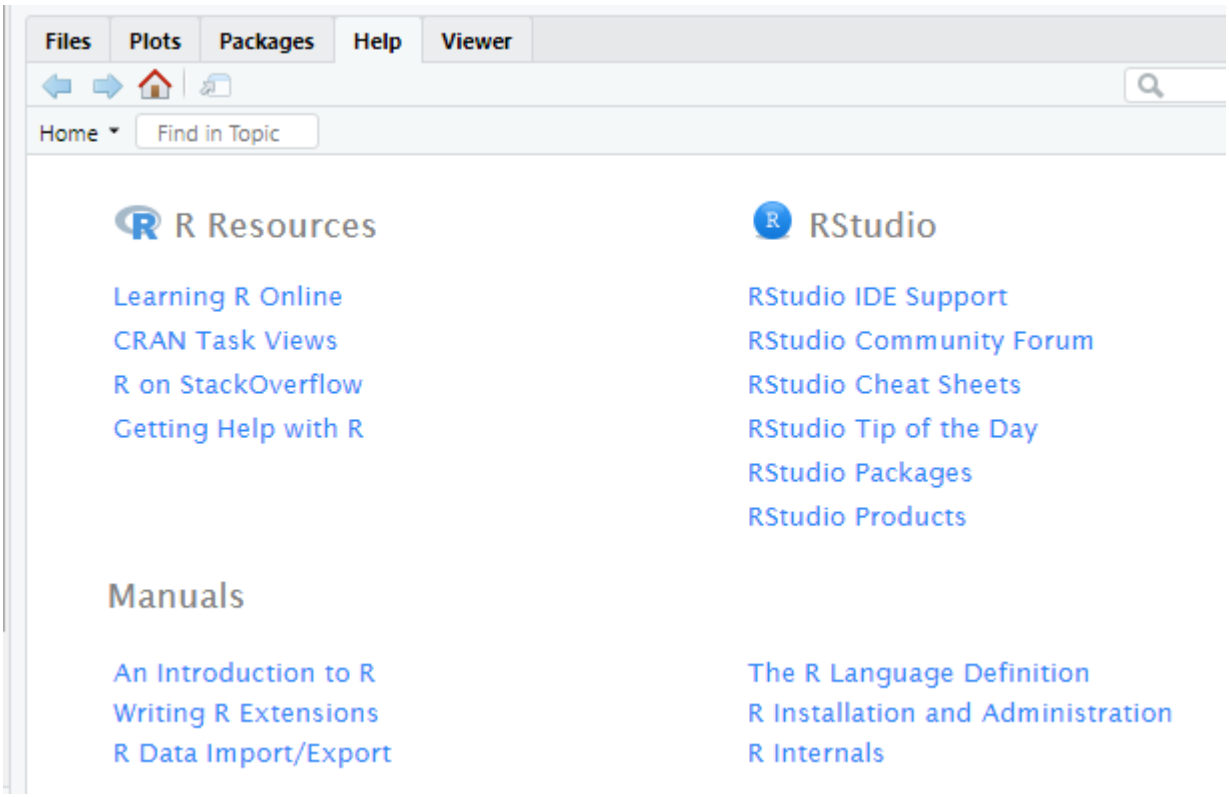


Figure 1: *Figure 1*

3 Manuals

The R distribution also comes with a lot of manuals. In RStudio, you can find the manuals in *Help* window, see window 4 of figure 1.



3.1 R help

Several hundreds of help pages, for all functions of R, are available online. They can be accessed by typing `help(NAME)` or `?NAME` at the console (see field 3 of figure 1) where `NAME` is the name of the function help is sought for.

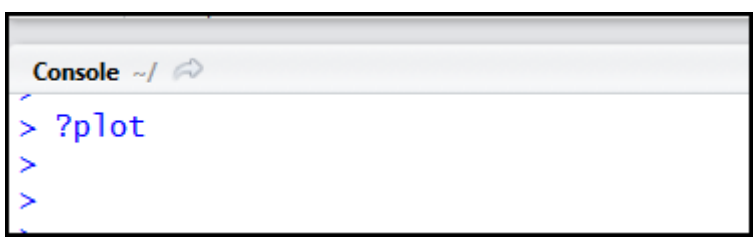
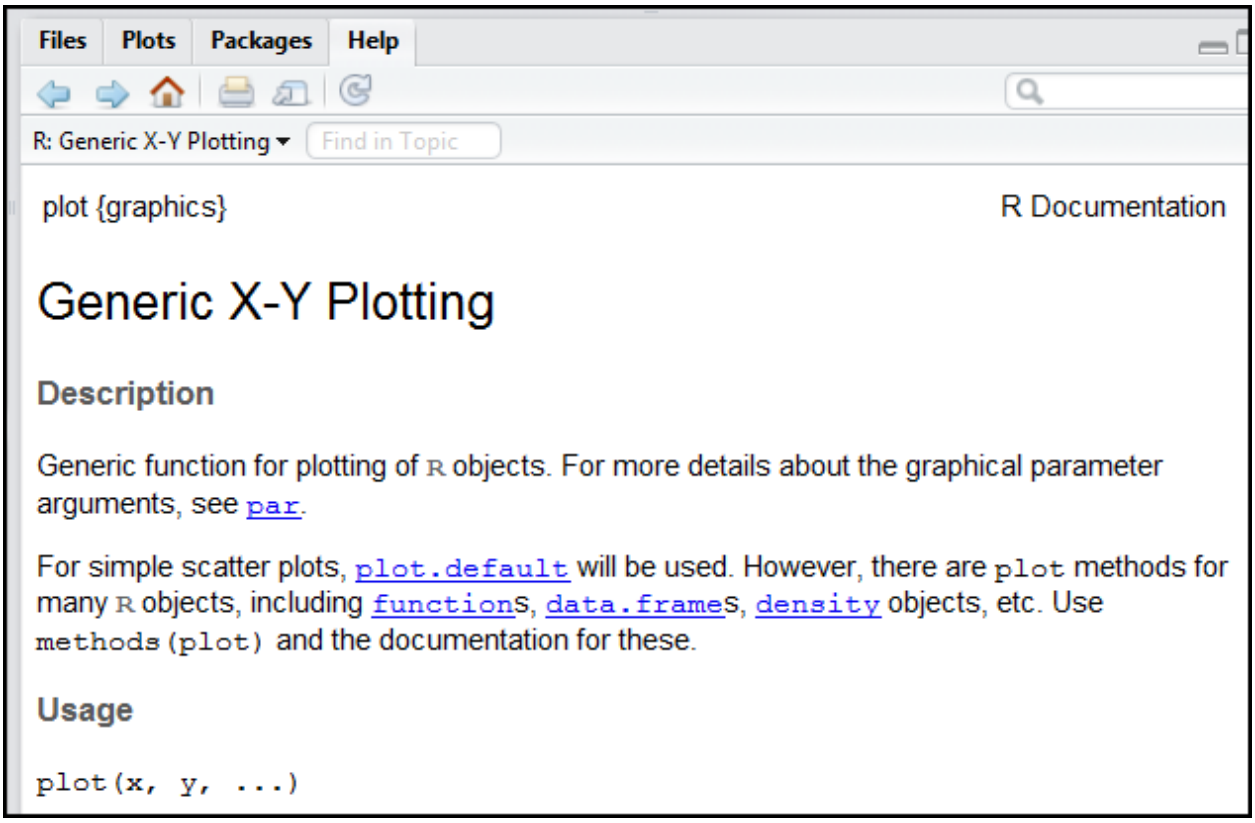
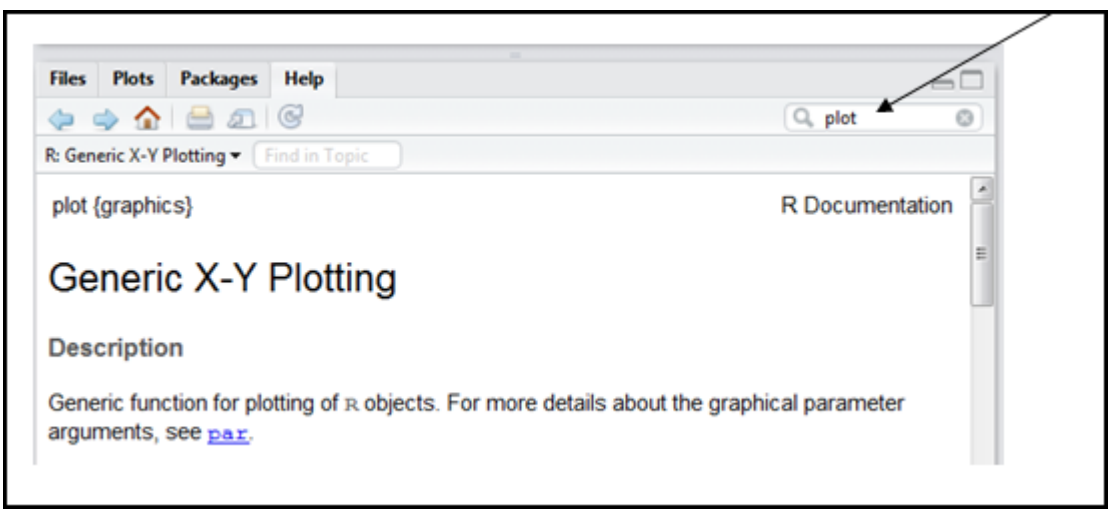


Figure 2: *Example for accessing the help pages of the function* `plot`

As a result, the corresponding web-page from the help-sever is displayed field 4 of figure 1:



You can get the same result by using the search option in the *Help* window in field 4 of figure 1:



3.2 Statistical methods with R

- Classical statistical tests
- Generalized linear models
- Multivariate statistics
- Linear and non-linear mixed effects models
- Robust statistics
- Survival analysis
- Classification and regression trees

- Neural networks
  And many others...

4 Remarks

1. **R is case sensitive**
2. Commands are separated by a semi-colon (`;`) or by a new line.
3. **Comments** can be put anywhere, starting with a **hash mark** (`#`). Everything to the end of the line is a comment.
4. **Assign** a value to an object by `<-` or `=`.

Example:

```
x <- 5
x
```

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

```
x = 5
x
```

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

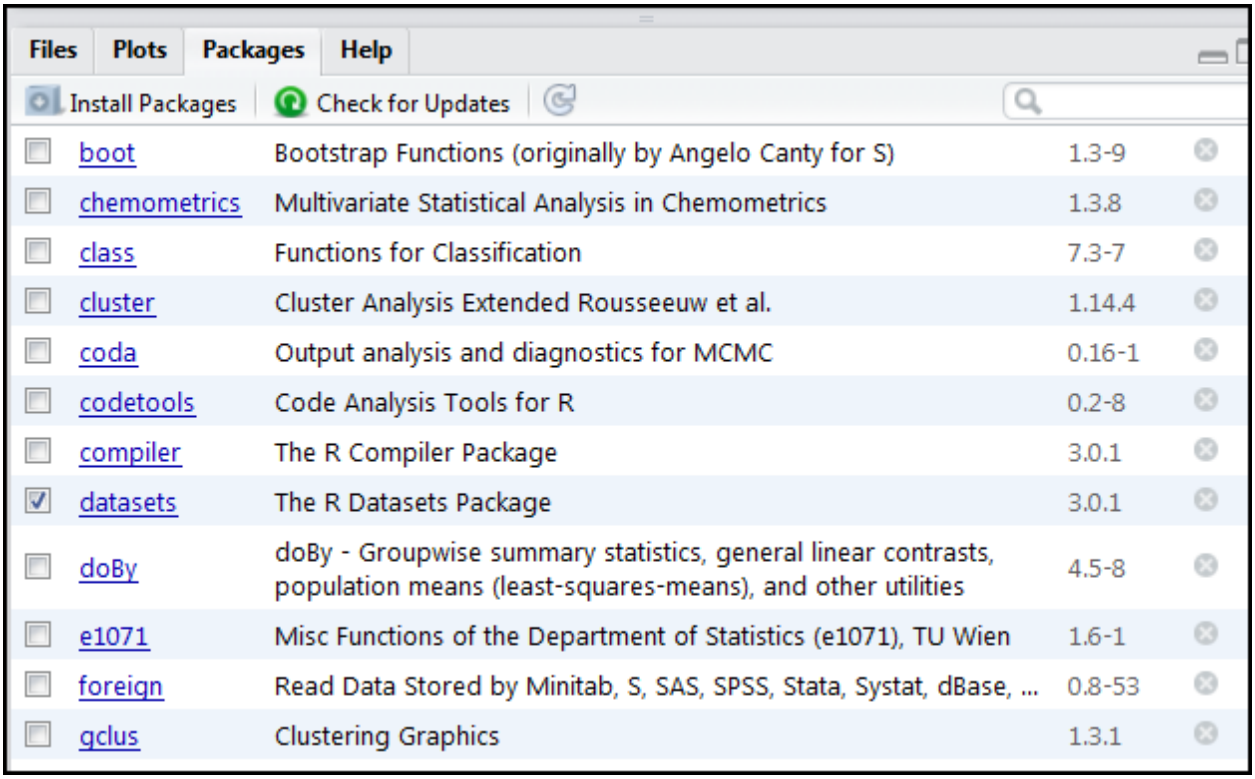
5 Add-on packages in R

It is important to distinguish the following about add-on packages in R:

- To **install** a package: To download a package
- To **load** a package: To activate all its functions

Every package has to be installed only once (if the same computer is used).

Remark: R comes with a limited list of packages.



5.1 Install package

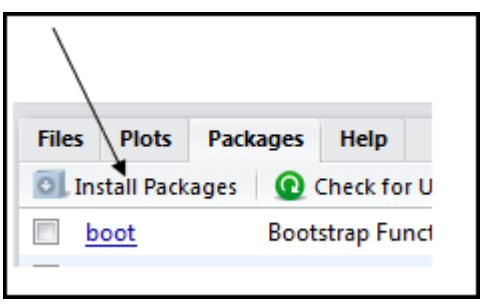
If you need to install (download) an extra package which is not yet available on your system, use the function `install.packages()`.

For instance, to install package `BayesTree`:

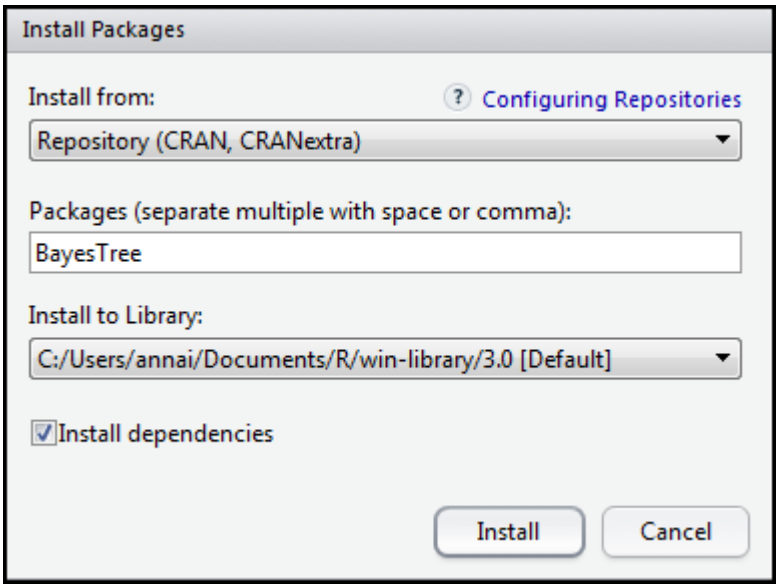
- Use the R code to install the package `BayesTree`:

```
install.packages("BayesTree")
```

- Use "Packages" window to install package `BayesTree`
  In the "Packages" window, click on "Install Packages" button:



Then, complete the dialog window as follows and click "Install"



Remark: Do not forget to load an installed package before you can use it (see next topic)

5.2 Load package

How can add-on packages be loaded?

5.2.1 Load packages by using R code

You can load the installed package `BayesTree` by typing in the console:

```
library(BayesTree)
```

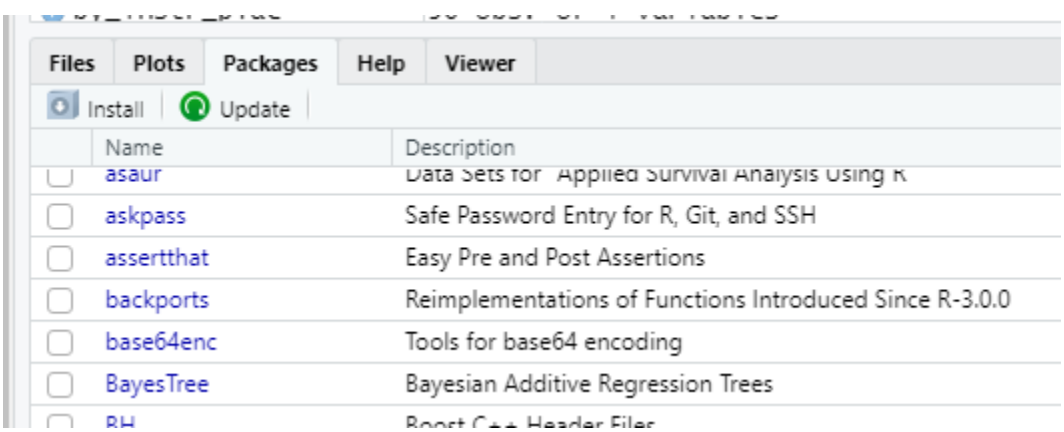
Remark: To find out which functions the package provides use

```
library(help = BayesTree)
help(package = BayesTree)
```

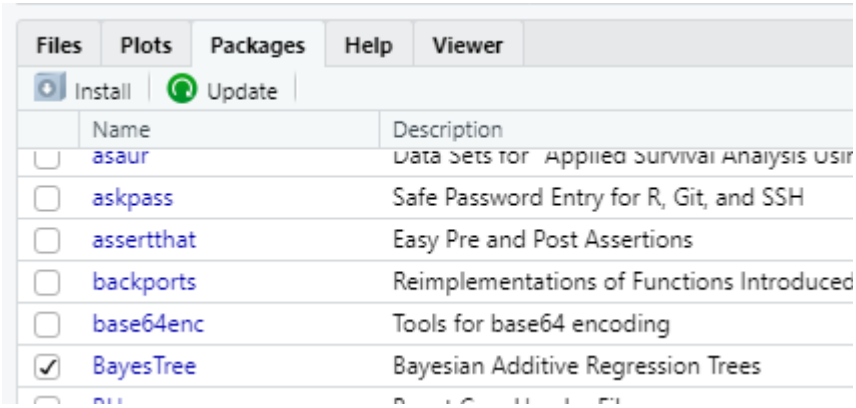
You will obtain in the help window the corresponding web-page from the R help-server.

5.2.2 Load packages by using "Packages" window

To find out which additional packages are installed on your system, check the list in the window "Packages" (see field 4 of figure 1). Here, the loaded packages are defined with a ticked box.



You can load the installed package `BayesTree` by ticking the box with this package:



Remark:

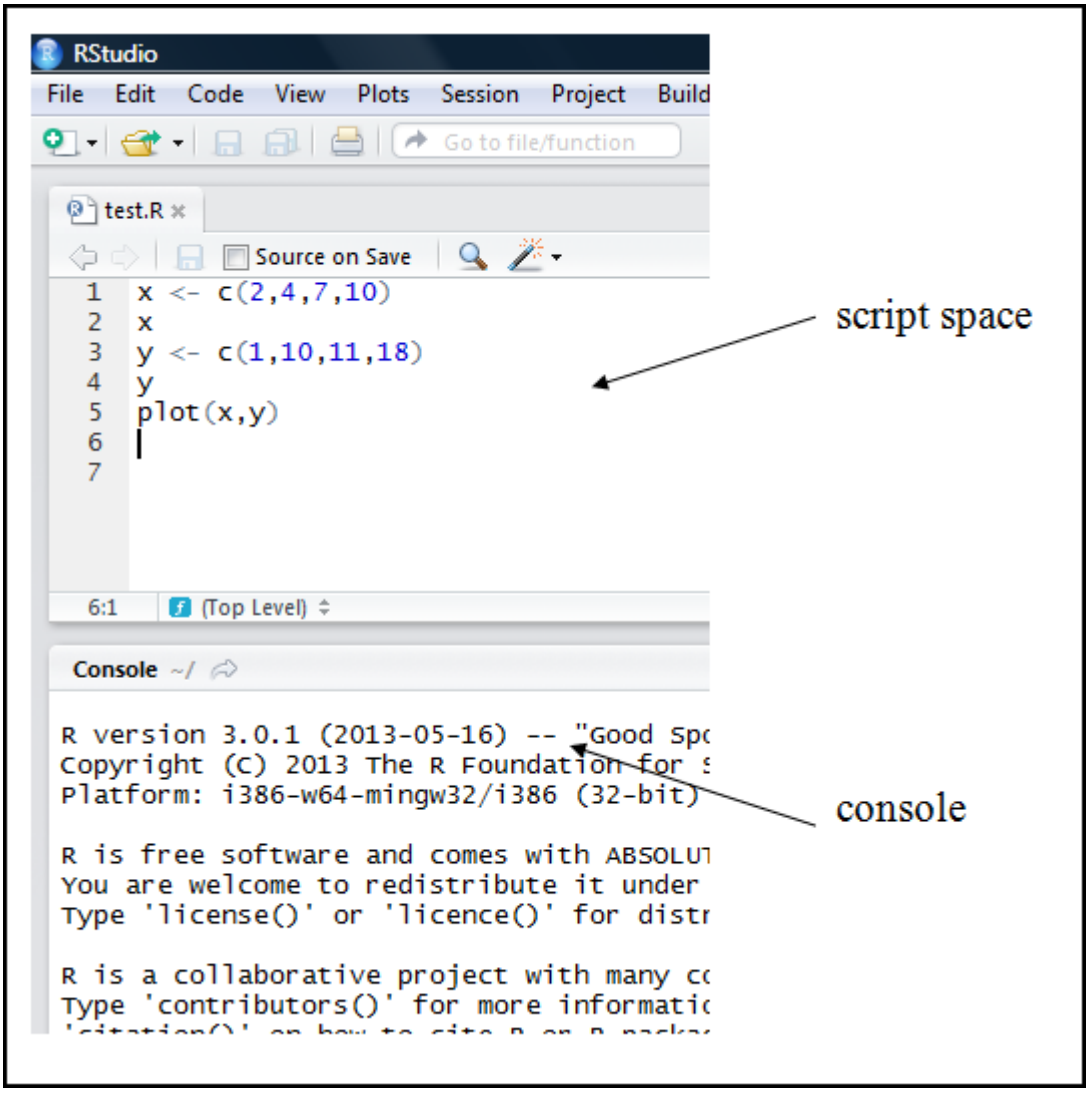
1. To ask for the contents of the package, simply click the underlined name of the package in the "Packages" window.
2. If you want to make it clear what package a function comes from, we will use the package name followed by the name of the function like: `dplyr::filter()`
   `dplyr` is the name of the package, `filter` is the name of a function in that package.

6 Exercise:

1. Check which packages are installed on your system.
2. Install and load the package `robustreg`
3. Check the contents of this package
4. Check which packages are loaded in your system.
   Hint: Use function `search()`

7 Writing scripts

Up to now we used for the syntax only the "Console" field. From now on, we will use 2 fields:



- **Script space** – contains the commands and comments, it can be stored as a file (see later more);
- **Console** – contains the executed commands and output. When you close RStudio, all syntax that was typed in the console will be lost!

A script is a list of commands in a file. Useful features of the script are:

- You can execute the code several times.
- You can include any comments (lines that begin with the `#` character) to remember or to inform others what the script is doing and why.

7.1 Common exercise

Here is an example step-by-step description of how to create and run a simple script that produces a plot.

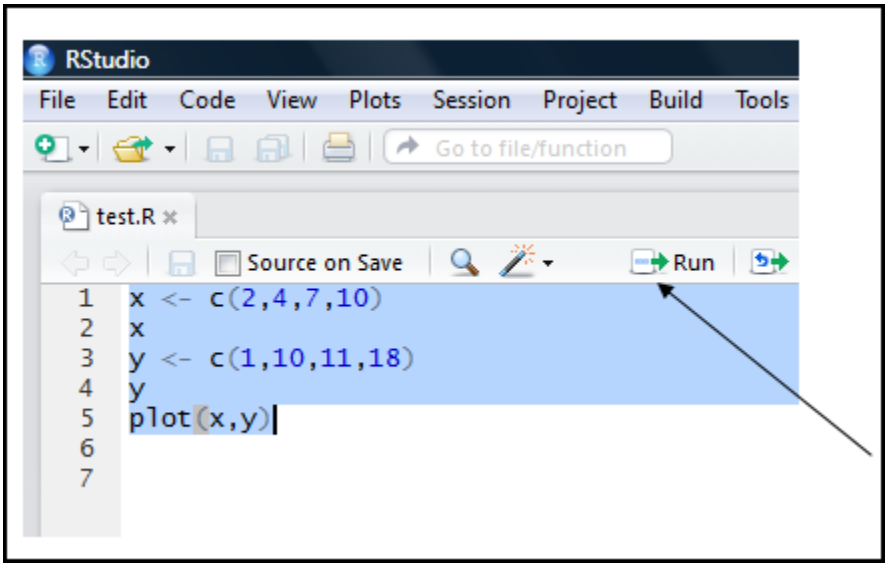
1. Open in RStudio a text editor: `File > New File > R Script`
2. Type in following lines

```
x <- c(2, 4, 7, 10)
x
y <- c(1, 10, 11, 18)
y
```

```
plot(x, y)
```

Note: `c()` is a function for creating a vector, `plot()` is the function for generating a scatter plot.

3. Save the file with the name `test.R` in a directory. Close this file.
4. In RStudio, select menu command
   `File > Open File...`
5. In the file selection dialog, locate the file `test.R` that you just saved and select it.
6. Execute the commands of the script:
   - highlight
   - use the "run" button from the toolbar (or use the combination of the keys `Ctrl` + `Enter`)



7. Examine the output.

Remark: When writing scripts, be sure to have the list of packages at the top of the script. Then it is easy to see which packages you need to run the script.

8 Working directory

- You can check the working directory by using the function `getwd()`.

You will certainly have another result than below.

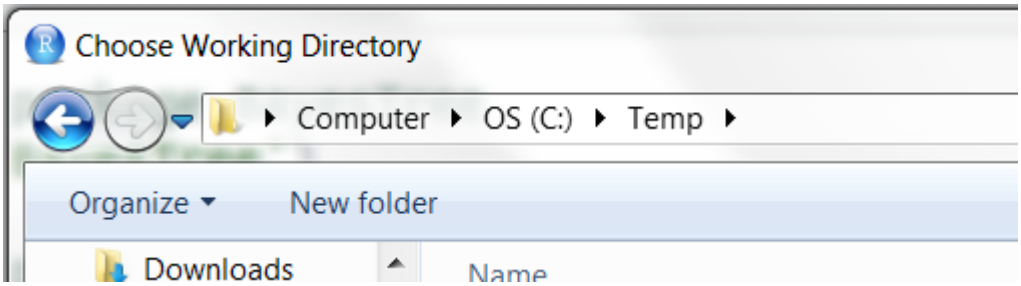
```
getwd()
```

```
## [1] "I:/AnCa/Data/Kursussen/Kursussen/R_software/2020-2021/notes_new/Chapter1"
```

Usually, R's default working directory is "C:/Users/User Name/Documents".

- You can change the working directory by using the function `setwd()`.
  For example: 'setwd("C:/Temp"). In this example, the working directory has been set to a folder *Temp*.

You can also use the menu Session in RStudio: `Session > Set Working Directory > Choose Directory...`



Remark:
`ls()`: list all objects in the current workspace
`rm(list = ls())`: removes all objects in the current workspace