

I. Working with R (exercises)

Data Science Lab, University of Copenhagen

August 2025

Getting started

- Make a new folder on your laptop for files associated to the course.
- Start RStudio.

Write commands at the prompt

1. Go to the R console (lower left window) and write a few commands, **one at a time**. You could for example try these commands:

```
6*12
x <- 100
x + 7
```

Notice how a new object, *x*, appears in the Global Environment window (upper right window) and can be used for new computations.

Commands written at the prompt are not saved for later use! It is fine to write commands that should never be used again at the prompt, and it is fine “to play at the prompt”, but in general you must organize your commands in R scripts (or R Markdown files, which will be introduced in Lesson IV).

Working with R programs

One option is to write your commands in a so-called R script. An R script is just a file with R commands and it usually has file extension *.R*.

2. Start by opening a new R script (*File* → *New File* → *R Script*).
3. Write a command in the file, similar to one of those from above. Click *Run* - or even easier - type *Ctrl and Enter*. Then the command is transferred to the prompt and executed, just as if you write the command at the prompt. Try with a new command in a new line.
4. Put a hashtag (#) before one of the commands and run it. Nothing happens! Hence, you can use hashtags for writing comments in your scripts.
5. Save the document in the folder you have designated for your R work (*File* → *Save As*) so that you can use it another time. When you save an R script it should be given the file extension *.R*. Thus, you might choose to call the file *solution1.R*.

Save your files often. Although very rarely, it happens that you ask R for something so weird that it shuts down, and then it is a pity to have lost your work.

Making some graphics

Let's also try making a few plots using the base graphics system in R (in *Presentation III* you will learn to make graphics using the *ggplot2*-package; this is the recommended approach).

6. First we need something to plot, that is a dataset. R has some built in datasets, which may be loaded using the command `data`. Insert the following two commands in your R script, and execute them (if you forgot how to do this, then see question 3 above) to learn about the *trees* dataset. Here the prefix `?` in the second command opens a help page in the lower-right window.

```
data("trees")
?trees
```

7. The dataset *trees* contains 31 observations of 3 variables (diameter, height and volume of black cherry trees). Insert the following two commands in your R script, and execute them to make two plots. Thereafter click “Plots” in the lower-right window, and use the arrows to toggle between the two plots you just made. What is the difference between the two plots?

```
plot(Volume~Height, data=trees)
plot(Volume~Height, data=trees, log="xy")
```

Shut down R

8. Close RStudio. If changes were made either to the R script (shown in the upper-left window) or to the workspace (called ‘Global Environment’ in the upper-right window) you will be prompted if you like to save those. Answer *Yes* to that; in particular it is important that you save your R scripts (or Markdown documents) as they contain all relevant commands to reproduce your output and plots. On the other hand, it is usually not necessary to save the objects that were generated, because they can be constructed with the relevant code.
9. Locate the R script that you saved in question 5 (and that presumably was re-saved in question 8); that would be the file *solution1.R* if you followed our name suggestion in question 5. Start RStudio again, and open the file (via the File menu). Check that you can run it again.

Remark: You can also start RStudio by double clicking on a file with *.R* (or *.Rmd*) extension. One advantage of this is that the workspace and the R history will be saved to the same folder as the R script when you later close RStudio, since the *working directory* will be set to the map where the file is located.

R packages

R is born with a lot of functionalities, but the enormous community of R users also contributes to R all the time by developing code and sharing it in R packages. An R package is simply a collection of R functions and/or datasets (including documentation). As of August 8, 2023, there are 19897 packages (1881 more than August 17, 2021) available at the CRAN repository (and there are many other repositories).

An R package should be *downloaded* and *loaded* before you can use its functionalities. You only have to download a package once (until you re-install R, or want to update the package), whereas you have to load it in every R session you want to use it. Let’s use the package **emmeans** as example.

10. Installation. There are several ways to do this, one of them is this one: Click *Tools* in the top menu bar, choose *CRAN* in the first drop-down menu, write **emmeans** in *Packages*, and click *Install*. A lot of red text is written in the console while the installation goes on. In the end, the package is installed.
11. Loading: Load the package with the command `library(emmeans)`. If everything went well, you should now be able to run the command **oranges**. This shows a dataset which is included in the package.

Getting help in R

Every R function comes with a help page, that gives a brief description of the function and describes its usage (input/arguments and output/value). Let’s use the function **median** as example. It is, no surprise, computing the median from a vector of numbers.

12. Try these commands:

```
x <- c(1,3,8,9,100,NA)
x
median(x)
```

The first command defines a vector with six elements, but where the last number is missing (NA = Not Available). Since the last number is missing, `median` returns NA. However, could we make `median` find the median of the remaining numbers. Perhaps the help page can help out!

13. Try the command

```
?median
```

Then the help page for `median` appears in the lower left window. If we read it carefully, then we realize that the extra argument (input) `na.rm` may help us. We therefore try this:

```
median(x, na.rm=TRUE)
```

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

Admittedly, R help pages are often quite difficult to read, but be aware that there are examples of commands in the bottom of each help page. For more complicated functions, these examples can be very useful while trying to get to know the function and its functionalities.

In order to use the help pages as above, you need to know the name of the function, which obviously may not be the case: You want to compute the median but have no idea what function to use. The best way to proceed is to **google** or use your favorite **chatbot**. If you google, then use “R whatever-you-want-to-search-for”, and you often get exactly what you need.

When working with R, you will get a lot of error messages. Some are easy to understand, and you will readily be able to fix the problems, while others... Again, try googling or try a chatbot. Copy the error message into google or to the chatbot, and you will often get useful help.

Finally, notice that there are a couple of **cheat sheets** available in RStudio: Click *Help* and then *Cheat sheets*.

Lessons learnt

- You must work with R scripts (or with R Markdown) such that you can save the work and return to it some other day. The course material is written in Markdown, but you can make the exercises with R scripts if you prefer - and at least until you learn how to use Markdown (see Lesson IV on day 2).
- Save your files often.
- If you forget to save your files before you close RStudio, then RStudio will prompt you if you want to save your work.
- Write comments to yourself along the way, so you can later remember why you did as you did. In R scripts this is done using hashtags (#).
- Google and chatbots are often very helpful.

End of exercise