I. Working with data in R (exercises)

Data Science Lab, University of Copenhagen

16 May, 2022

Getting started

- Start RStudio.
- Make a new *R-project*. There are a couple of ways to do this. You can make one in a brand new folder or make one in an already existing folder. We will create a new folder.
  Go to the menu bar and click *file* → *New project...* → *New Directory* → *New Project*. **NOW**, make sure you know where you place your folder. Name your directory (folder), then click *Browse* to place your new folder and project somewhere you can find it again.
- Lets check which working directory we are in. We do this with `getwd()`(you do not need to put anything in the parenthesis).
- *IF* you want to change your working directory you can do this with `setwd("path/my.folder")` or by clicking *Session* → *Set Working Directory* → *Choose Directory...*.

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 thee 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 by clicking *File* → *New File* → *R Script*. (You can also make new script simply by clicking the *icon with the green plus* at the top most right of the Rstudio editor).
3. Write a command in the file, similar to one of those from above. Click *Run* - or even easier - type *Ctrl + 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. 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.

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 20, 2020, there are 16153 packaged available at the CRAN repository (and there are many other repositories).

An R package needs be *installed* and *loaded* before you can use its functionalities. You only have to download and install a package once (until you re-install R), whereas you have to load it in every R session you want to use it. As an example, let's install the package `emmeans`.

6. Choose one of the two installation methods:

a. Using the command line:

```
install.package("emmeans")
```

Note that for this approach you need to know the name of the package and spell it correctly, including capitalization!

b. Using the graphical interface:

Look at lower left of your Rstudio window where you have a window with several tabs. Click on *Packages*. You will see a list of your currently installed packages and their versions. To install `emmeans`, click on *Install* and start to type the name. You will notice a drop down list appears from which you can select the correct package. This is useful if you are not quite sure of the correct spelling.

A lot of red text will be written in the console while the installation goes on. This usually does not mean there was a problem, unless the text reads 'error' or 'exit'. In the end, the package is installed, or you will see an explanation of what went wrong.

Notice - if you have a SCIENCE-PC: There is sometimes problems with installation of packages on SCIENCE-PC's, because R tries to install files to a place, where you don't have permissions to save and edit files. There is a work-around that (sometimes) works, so talk to us if you have the problem.

7. Loading: Load the package you just installed with the command `library(emmeans)`. If everything went well, you should now able to run the command `oranges`. This shows a dataset which in included in the package.

Basic R commands

We will now have a look at basic commands in R that you just learned about. We will use an in-build dataset of R called `trees`.

8. To start with, load the `tidyverse` package.

```
load(tidyverse)
```

9. Now, load the `trees` dataset by copying and executing the following commands:

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

10. Check what information is contained in the `trees` dataset by calling the `summary` function on it:

```
summary(trees)
```

11. How many trees do we have data for?
12. Now, extract the column called `volume` from the dataset and assign it to a new variable called `volume_col`.
13. Display the volume of the first 10 trees.
14. Find the minimum, maximum and mean volume from the column you just extracted. Does it match what was stated in the `summary`?
15. What class of datastructure is `trees`? Make it into a tibble.

Making some graphics

Next, let's also try to make some plots using the base graphics system in R (in *Presentation III* you will learn to make graphics using the *ggplot2*-package). We will continue to use the `trees` dataset.

16. 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.

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

17. The plots will appear in the *"Plots"* window on the lower right. Click on *"Plots"* if you still have the *"Help"* tab selected. Use the blue arrows to toggle between the two plots you just made. What is the difference between the two plots?

Shut down R

18. 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. In ERDA you can choose to open a new R session (or you can close the window if you don't want to continue working).
19. 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 - if you work locally on your laptop (not ERDA): 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.

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.

20. 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!

21. Look at the help for the `median` function:

```
?median
```

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)
```

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: Google! Use "R whatever-you-want-to-search-for", and you often get exactly what you need.

While 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, the best answer is: Google! Copy the error message into Google, and you will often find help.

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 will have learnt how to use Markdown.
- 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 (#).