Introduction to R and RStudio

true true true

11/20/2025

Goal of this course

- Gentle introduction to:
  - Programming languages
  - R for data science
- Provide a foundation for the advanced R course

Why do we care about R?

Why programming languages?

- Faster
- Easier (once you know it)
- Communication: beautiful plots/interactive
- Flexibly (new methods)
- Reproducibility! (open an old Excel file and try to understand what you did)

Why R:

- Designed for statistics/data science (=easy pipelines)
- Huge community of users (=support)
- Integration between libraries (=everything just works)

R for data science

What is R / RStudio?

Communicating with the comptuer

R (~English)

- Programming language, allows you to communicate with the computer

RStudio Integrated Development Environment (~Word)

- Makes using R as effective and efficient as possible:
  - Code editor
  - Nice extras: syntax highlighting; code completion; file explorer; help

R packages (~tabs in Word)

- Extend the functionalities of base R
- You can install and use new packages:

```r
install.packages("ggplot2") # Install new package (you only need to do it once)
library(ggplot2) # Load the package
```

RStudio

RStudio

Source editor Write your R code (load data, clean it, model it, etc)

Environment/Workplace All the variables that you have defined

Files File explorer, find your files.

Help Get information about code (super useful!)

Console Write R code (not recommended at this point) and see the output of your R scripts

Plots See the plots, and export it

History History of all the code you have run.

Packages All packages that you have loaded (I don't recommend loading/unloading packages this way)

Terminal Run commands on your terminal (this is not R, you won't need to use this)

Basic units of RStudio

Run pieces of code

View your data/objects

Create a new file: R Scripts and Quarto Documents

Paths & working directory

Paths and working directory (folder)

- Your computer finds files by their path:
  - "/Users/yourname/Desktop/somefile.csv" (Mac/Linux)
  - "C:\Users\yourname\Desktop\somefile.csv" (Windows)
- Both in Unix and Windows:
  - Always use "/" (R will convert it)

    - “\” is a “escape character”, with a special meaning
- Paths can be:
    - Absolute (e.g., the ones above). They are defined from the root directory, the top-most directory.
    - Relative. They are defined from the working directory
        - e.g. “data/somefile.csv” –> in our working directory, find the folder “data”, and inside, the file “somefile.csv”
        - It makes possible to use the code in different computers.

Paths and working directory (cont)

Important shortcuts for relative paths:

- “`../`somefile.csv”: find “somefile.csv” one level down
- “`../../`somefile.csv”: find “somefile.csv” two levels down
- “`./`somefile.csv”: find “somefile.csv” in the current level (not so useful, it is identical to “somefile.csv”)
- “`~/`somefile.csv”: find “somefile.csv” in your home directory

Changing the working directory with RStudio:

- “Session (top menu) -> Set Working directories”
- “Files (RStudio unit) -> More –> Set as working directory”
- On a notebook –> Working directory = directory of the notebook

Type of R documents

R-scripts (.R)

- Text file with R code (you can open it in any text editor)
- Working directory:
    - If not associated to a project: your home directory (e.g. “/Users/javiergb”)
    - If associated to a project: the project directory
- Output in the console/plots menus

Rstudio-projects (.Rproj)

- Contains a .Rproj file within the directory, with project options. More info on setting up projects
- Working directory: the project directory where the .Rproj file resides.

R-notebooks (.Rmd) and Quarto documents (.qmd)

- Markdown files (like this one!) – combines chunks of texts and code
- Working directory: the directory where the document is located
- Outputs directly in the editor. Can be knitted to HTML/PDF/Word
- Most useful for data science

How does R work

Variables

- Tell the computer to save an `object` (a number, a string, a spreadsheet) with a name.

- Creating variables in R is very straightforward:

    - you just use the assignment operator

```
<-
```

- For example, if you assign the value `100` (an element) to variable `a`, you would type

```
a <- 100
print(a)
```

```
## [1] 100
```

Code

- Instructions to modify variables
- Can be organized in functions: blocks of code that take some input and return some output

```
sum(a, 1)
```

```
## [1] 101
```

```
a |> sum(1)
```

```
## [1] 101
```

Objects

- Everything is an object in R, and can be assigned to a variable name

Basic data types (elements)

- `character`: "some text"
- `numeric`: e.g., 2.1
- `integer`: e.g., 2L
- `logical`: TRUE/FALSE
- `factor`: e.g., factor("amsterdam")

Basic data structures

- Consist of data types and functions to transform them
  - `vector`: c(2, 4, 2)
  - `list`: list(first_col = 1, second = "a", third = TRUE)
  - `matrix`: matrix(c(4, 4, 4, 4), nrow = 2, ncol = 2)
  - `data.frame`: The most important ~ spreadsheet

The help

- Everything that is published on the Comprehensive `R` Archive Network (CRAN) and is aimed at `R` users, must be accompanied by a help file.

- If you know the name of the function that performs an operation, e.g. `anova()`, then you just type `?anova` or `help(anova)` in the console, or use the "Help" menu.

- If you do not know the name of the function: type `??` followed by your search criterion. For example `??anova` returns a list of all help pages that contain the word 'anova'

- Alternatively, the internet will tell you almost everything you'd like to know and sites such as http://www.stackoverflow.com and http://www.stackexchange.com, as well as `Google` and `LLM` can be of tremendous help.

  - If you google `R` related issues; use 'R:' as a prefix in your search term
  - On social media, the #RStats is often used to discuss R

Calling objects

- You just use type the name you have given to the object
- For example, we assigned the value `100` to object `a`.

```
a <- 100
```

To call object `a`, we would type

```
a
```

```
## [1] 100
```

Using functions

```
# This is a comment, it won't be read by R
student_number <- 4
paste("The number of students is: ", student_number, sep = " ")
```

```
## [1] "The number of students is:  4"
```

```
#sep can be any character, or "\n" (newline), "\t" (tab),
```

```
vec <- c(1, 2, 3, 4)
mean(vec)
```

```
## [1] 2.5
```

```
mean(1:4)
```

```
## [1] 2.5
```

---

2. Using packages

```
# Install a package to read `csv` files
install.packages("readr")
```

```
# Loading the package, usually at the top of the R script
library(readr)
```

```
# Using the readr library
data <- readr::read_csv("../common_datasets/dataset_boys.csv", col_select = c("age","hgt"))
## Rows: 748 Columns: 2
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## dbl (2): age, hgt
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

Hint: the `readr::` is optional when the package is loaded

2. Data analysis

```
# Summary statistics
summary(data)
##       age              hgt
##  Min.   : 0.035   Min.   : 50.00
##  1st Qu.: 1.581   1st Qu.: 84.88
##  Median :10.505   Median :147.30
##  Mean   : 9.159   Mean   :132.15
##  3rd Qu.:15.267   3rd Qu.:175.22
##  Max.   :21.177   Max.   :198.00
##                   NA's   :20
```

Practical 1

Goal: Get used to RStudio using R as a calculator, and install one library

1. Create an R script
2. Create an R project
3. Create a Quarto document