

R Faculty Workshop

Workshop overview

Day 1: Intro to R

Intro to R

Some notes about R and its ecosystem in general

Basics of writing R code

Syntax, data types, assignment, functions

Working with data

Importing data, cleaning, graphs, analyses

Day 2: Teaching with R

General thoughts on teaching with R

Pedagogy, grading, generative AI

Posit Cloud

A web-based version of RStudio

Quarto

An alternative document format with advantages for teaching

Example course materials

Examples of how I use R and these additional tools in Statistics and Labs

1 Introduction to R and RStudio

1.1 Why R?

R is a coding language specialized for statistical computing and data analysis. It is free and open-source (though there is a cloud-based version which can be paid for and can have advantages especially in the classroom).

Some of R's capabilities:

- Import and create data files in various formats
- Clean and organize data
- Analyze and visualize the data
- Communicate the results in various formats (pdf research paper, website, presentation slides)
- Generate random data, execute functions repeatedly, useful for simulations, bootstrapping etc
- Other programmatic tasks, e.g. web-scraping, using APIs

1.2 The general workflow

1.2.1 Separation of data and code

It might seem daunting to learn R if you have no experience with coding, but the basic idea is that you have some data, like you are familiar with from a regular Excel or Google Sheets spreadsheet, and you perform operations on your data using functions a lot like you would in Excel/Sheets. For example, you might compute an average in Sheets by typing `=AVERAGE(A1:A10)`. In R you might type `mean(my_data$column_a)`. The specifics of the function names are different, but the basic idea is the same.

A major difference between working with data in Excel vs. R is the separation of data from code. Rather than writing functions to manipulate or analyze data directly in your spreadsheet,

code is written in a separate code file, which references **but does not modify** the source data file (unless you tell it to).

Excel Spreadsheet

A
1
2
3
4
5
=AVERAGE(A2:A6)



R Data

A
1
2
3
4
5

R Code

```
mean(data$A)
```

```
[1] 3
```

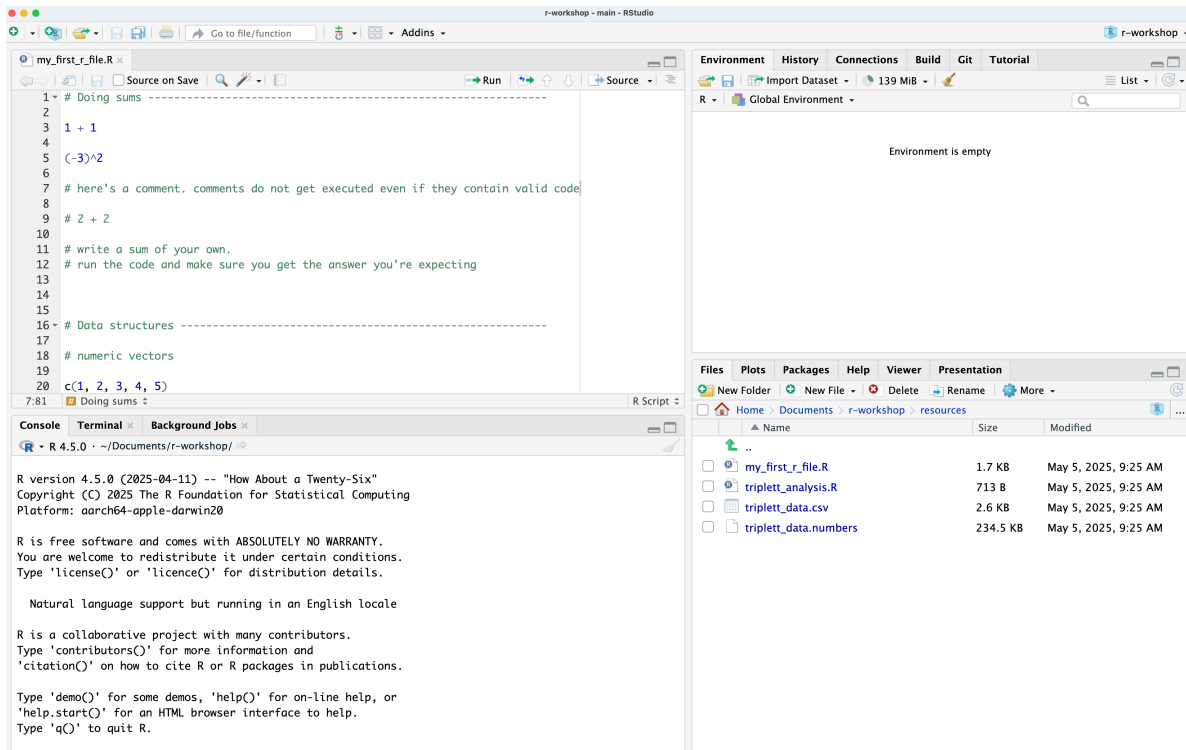
```
sum(data$A)
```

```
[1] 15
```

```
sd(data$A)
```

```
[1] 1.581139
```

1.2.2 RStudio Interface



RStudio is the interface we'll use to write and run R code and see its output. The basic interface has 4 panels, each with a few tabs:

- Top-left: Code editor / data viewer
 - Open, edit, and save code documents
 - Execute code within files
 - View data
 - You can have multiple 'tabs' open at once,
- Bottom-left: R console
 - You can type code directly and run it by pressing enter.
 - You won't be saving your code as a document like when you type in in the editor, so this is useful for testing something simple out
- Top-right: Environment
 - As you execute code you may be creating objects like sets of numbers or data.frames. Those objects will appear here.
 - You can click the name of some objects, like data.frames, and it will open a view of the data as a tab in the editor pane

- Bottom-right: Files/folders, Plots, Viewer, help window
 - You can navigate the file tree
 - And get Help with functions
 - As well as seeing plots and other kinds of output

1.3 Additional packages

The R language has many functions built in. Generally speaking, you can find a way to do pretty much anything you would like to do using just 'base' R.

However there are many common tasks that are a bit tedious or unintuitive to do using base R. One of R's strengths is how extensible it is: anyone can write their own functions, turn the code into an R package, and make that package available to other R users.

1.3.1 Tidyverse



The tidyverse package is a container for multiple individual packages. The whole family of tidyverse packages are written with a consistent syntax and logic, and are widely used for data analysis. `readr` handles importing data, `dplyr` and `tidyr` have many functions for data cleaning and manipulation, `stringr`, `lubridate`, and `forcats` are specialized for working with text, dates, and categorical variables respectively; `ggplot2` makes graphs; and `tidymodels` is for modelling.

1.3.2 Specialized analyses

The extended ecosystem includes packages specialized for almost any kind of analysis you can think of. To give a few examples...

- Structural equation modeling (`lavaan`)
- Meta-analysis (`metafor`)
- Linear mixed effects models (`lme4`, `simr`)
- Bootstrapping (`boot`)
- Bayesian models (`brms`, `rstanarm`)
- Network analyses (`igraph`, `ggraph`, `tidygraph`, `qgraph`, `bootnet`)
- Language analysis (`tidytext`, `quanteda`)
- Audio analysis (`tuneR`, `seewave`)
- Machine learning (`tidymodels`)

1.3.3 Additional capabilities

E.g. maps (`sf`, `leaflet`)

```
library(leaflet)

leaflet() |>
  addProviderTiles("NASAGIBS.ViirsEarthAtNight2012") |>
  addMarkers(lng = -73.96339268916061,
             lat = 40.80949994182454,
             popup = "Hello from Barnard!")
```