Introduction to Rstudio

Anders K. Krabberød

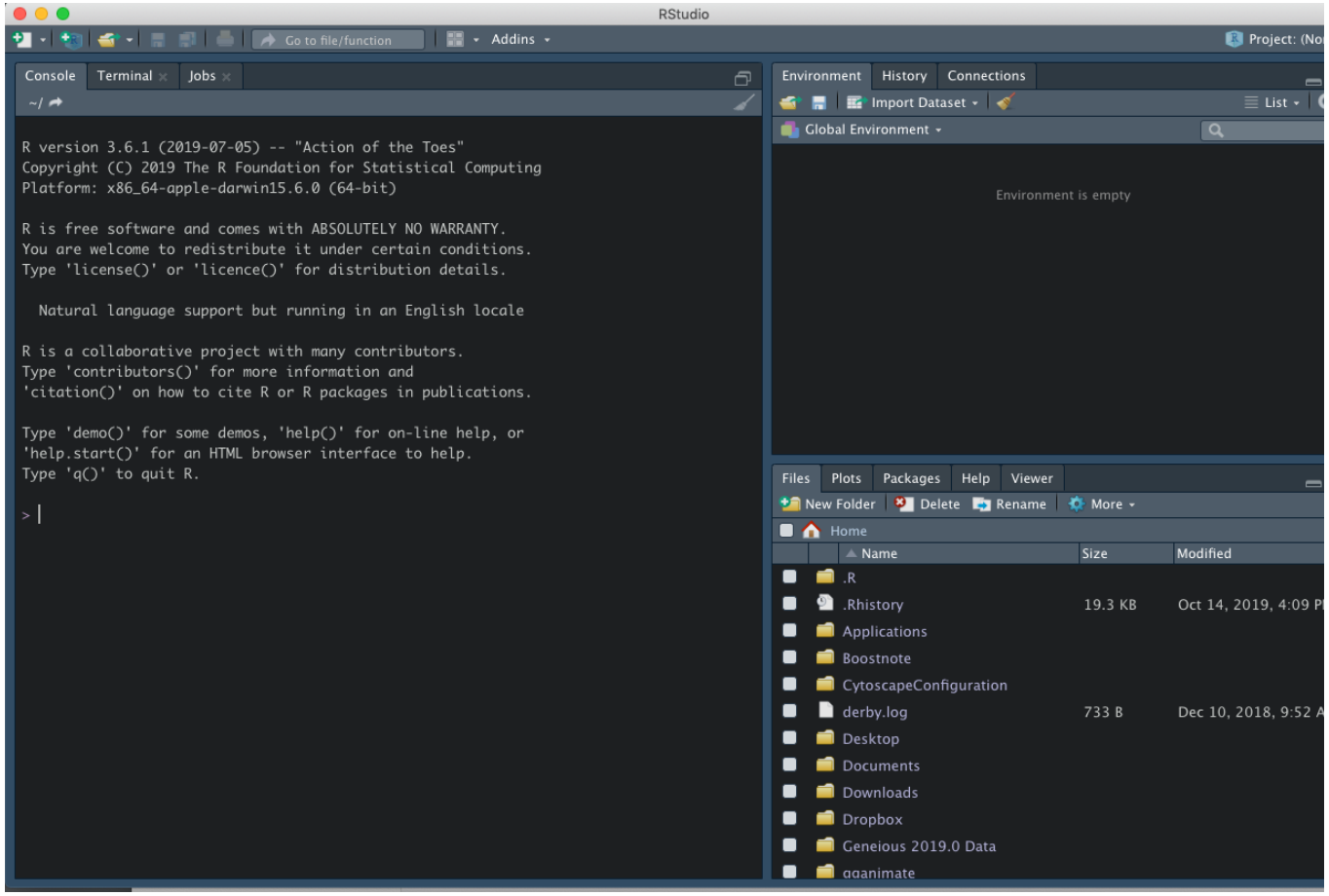
a.k.krabberod@ibv.uio.no

(remember GitHub)

- Github for the workshop with lectures, scripts, and examples:
- https://github.com/krabberod/OMG_bioinformatics_sessions_2022

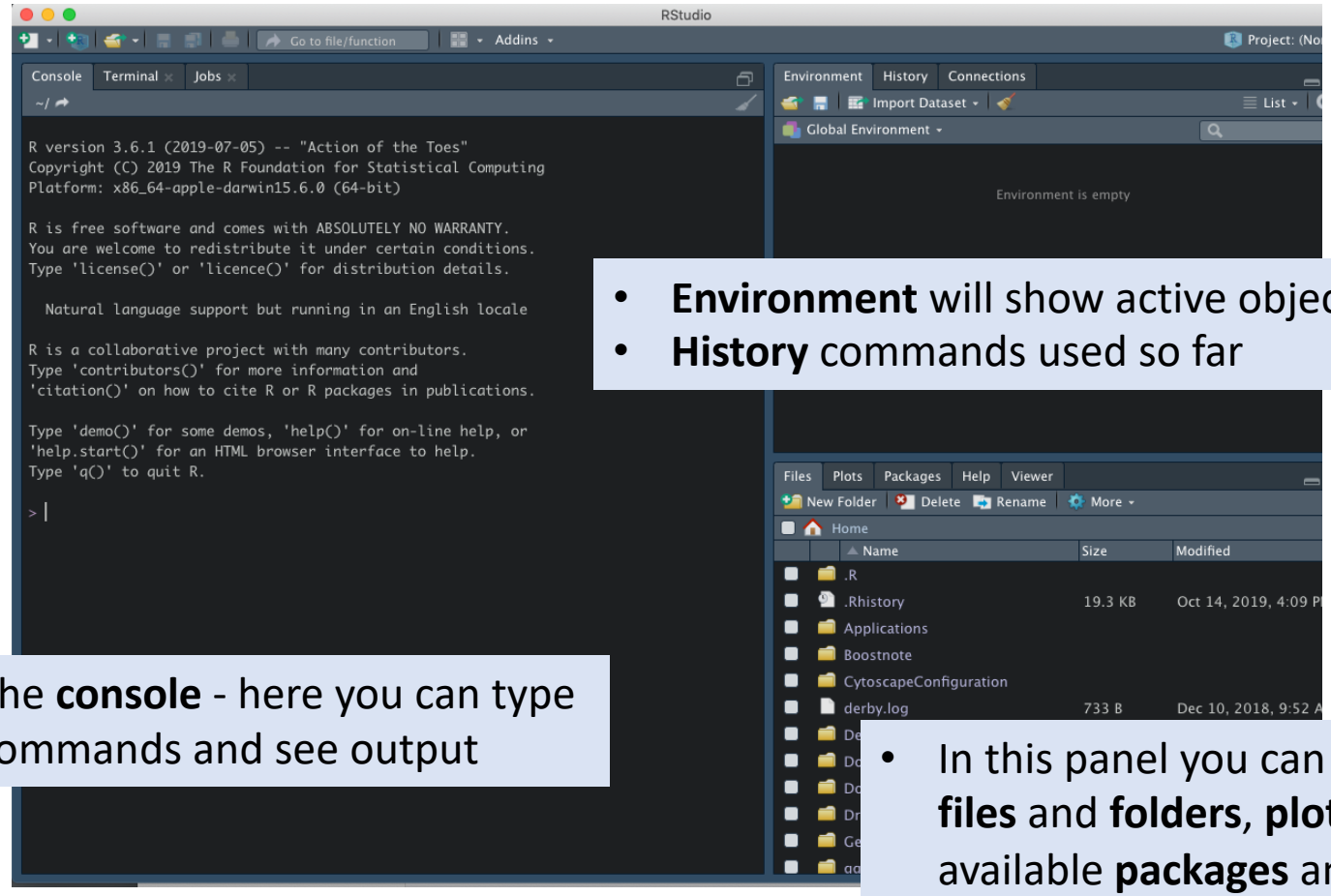
Rstudio

- A graphical user interface for running R (and other programming languages like Python, bash and more).
- Also has a file browser and a window with easy access to objects and variables.



Rstudio

- A graphical user interface for running R (and other programming languages like Python, bash and more).
- Also has a file browser and a window with easy access to objects and variables.

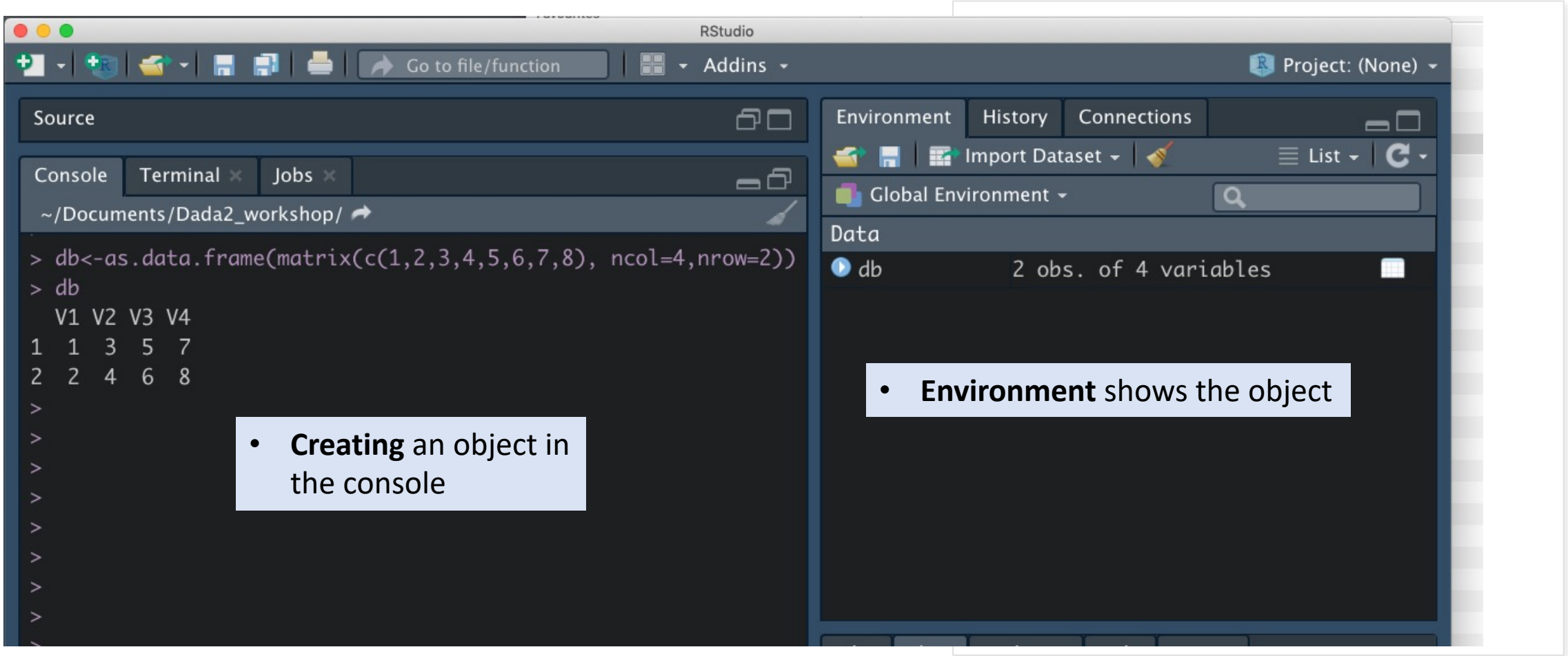


- **Environment** will show active objects
- **History** commands used so far

The **console** - here you can type commands and see output

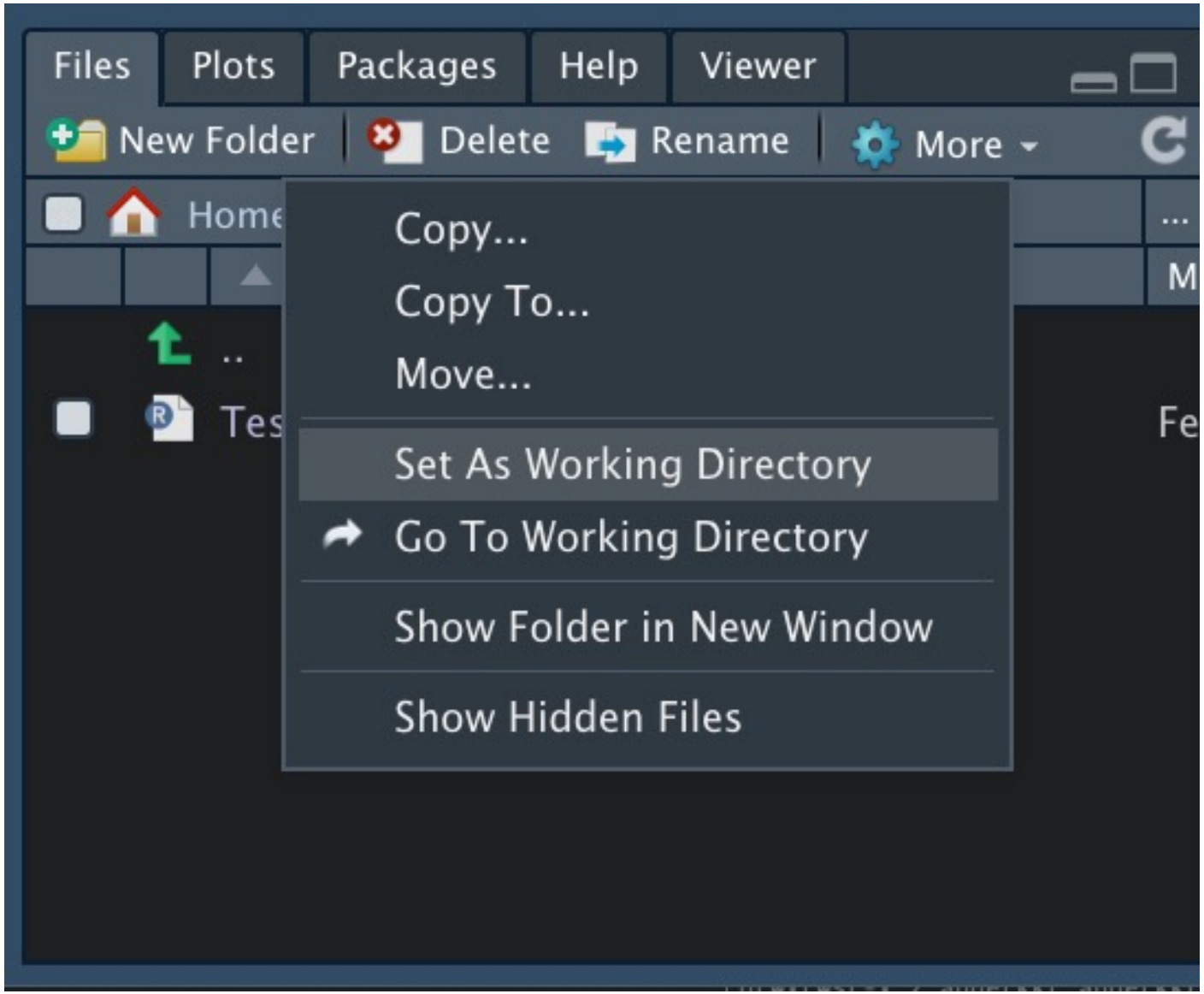
- In this panel you can find **files** and **folders**, **plots**, available **packages** and the help **screen**

Rstudio


RStudio
Go to file/function
Addins
Project: (None)
Source
Console
Terminal
Jobs
~/Documents/Dada2_workshop/
> db<-as.data.frame(matrix(c(1,2,3,4,5,6,7,8), ncol=4,nrow=2))
> db
V1 V2 V3 V4
1 1 3 5 7
2 2 4 6 8
Environment
History
Connections
Import Dataset
List
Global Environment
Data
db
2 obs. of 4 variables
• Creating an object in the console
• Environment shows the object

Setting working directory

- Setting the working directory will let you determine the folder where *Rstudio* will write output, print graphics and look for files to open.
- Navigate to correct folder under the “files” tab
- Click “Set As Working Directory” (under *More*)



Setting working directory

- Alternatively write

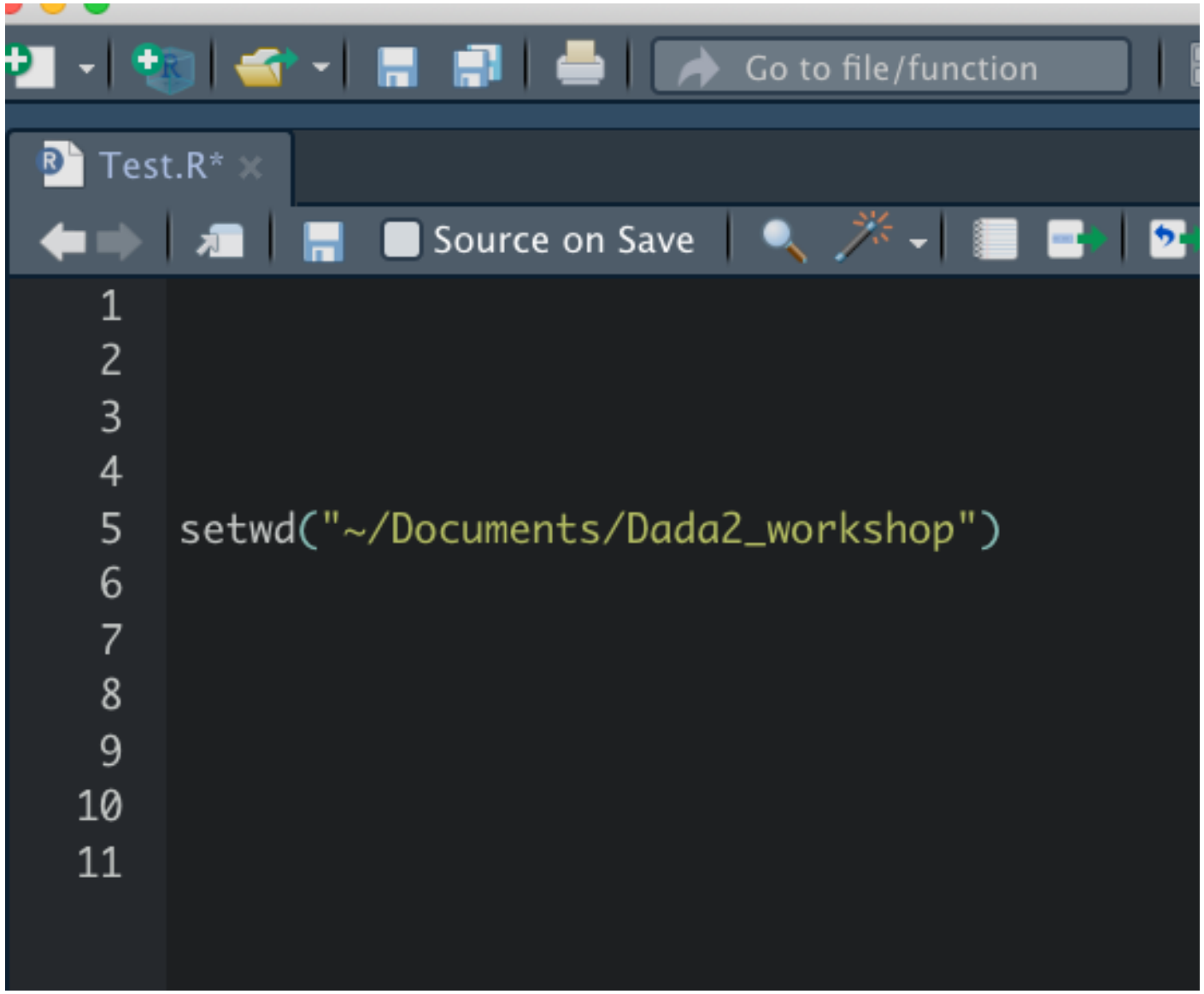
  MAC:

  ```
  setwd("~/path/to/my/folder")
  ```

  WINDOWS

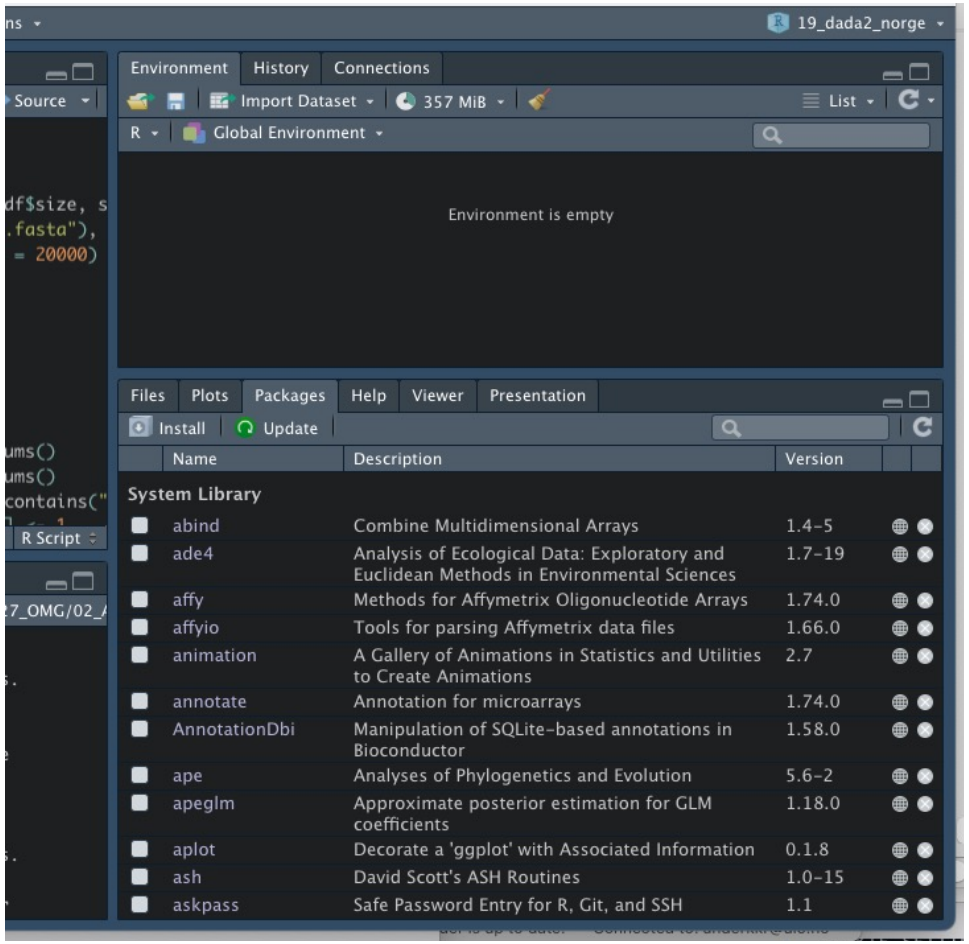
  ```
  setwd("C:/path/to/my/folder")
  ```

- This way you can include the path to the working directory as part of your script.



Installing packages

- Packages are a set of functions, compiled code, and sample data.
- When you start R only the default packages are loaded
- Other packages need to be installed and called explicitly to be utilized
- Packages can be found in three main repositories:
  - **CRAN**: Comprehensive R Archive Network(CRAN) is the official repository; it is a network of ftp and web servers maintained by the R community around the world. Reviewed and curated.
  - **Bioconductor** is a topic-specific repository, intended for open-source software for bioinformatics. Reviewed and curated.
  - **GitHub** is the most popular repository for open-source projects. *Not* reviewed and curated.



Installing packages

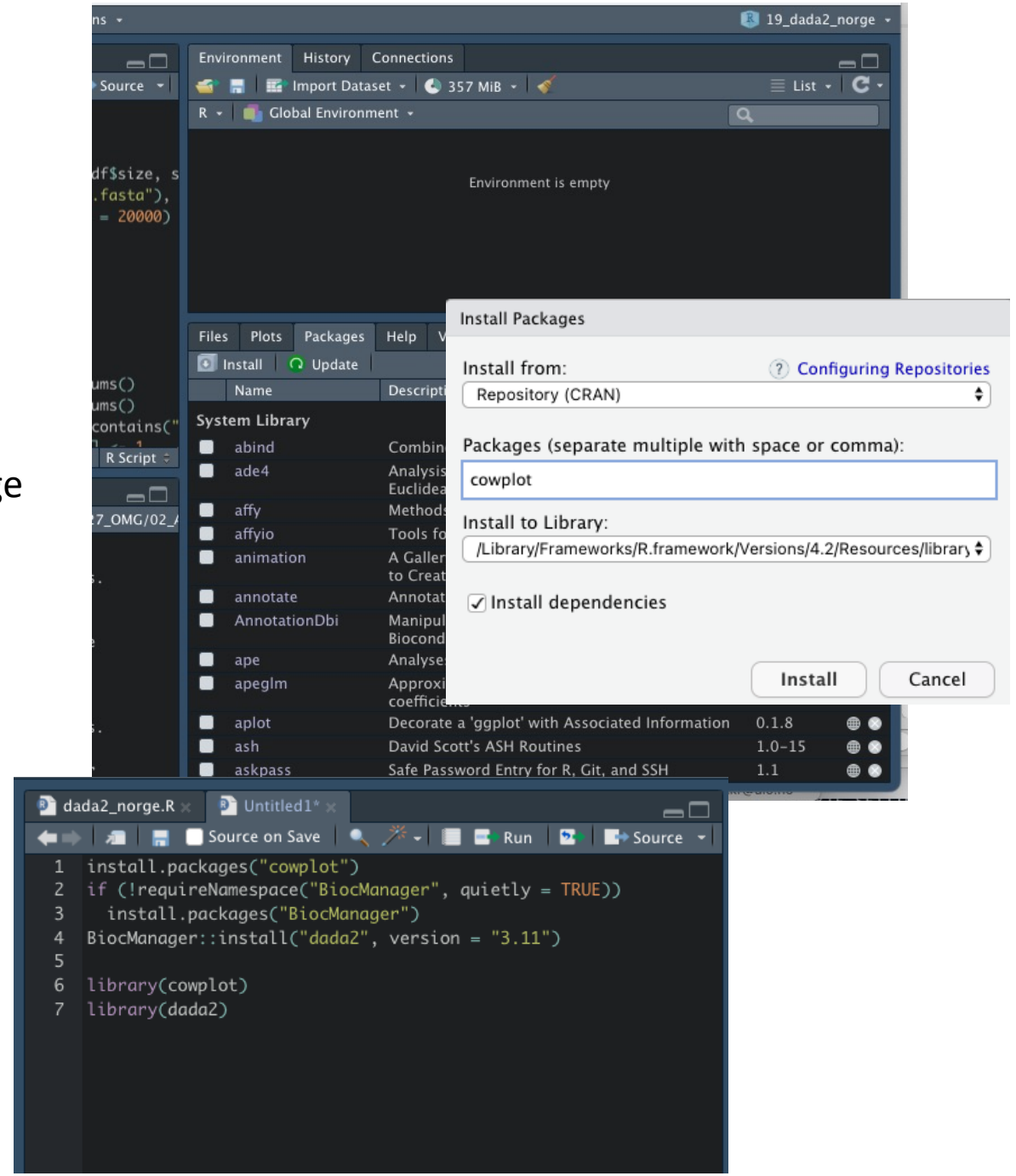
- Point-and-click available for CRAN (example for the package cowplot)
- Or for CRAN use the command :

```
install.packages("cowplot")
```

- For Bioconductor packages use (with dada2 as example):

```
if (!requireNamespace("BiocManager", quietly = TRUE))
    install.packages("BiocManager")
BiocManager::install("dada2", version = "3.11")
```

- The installation will take care of installing dependencies if necessary

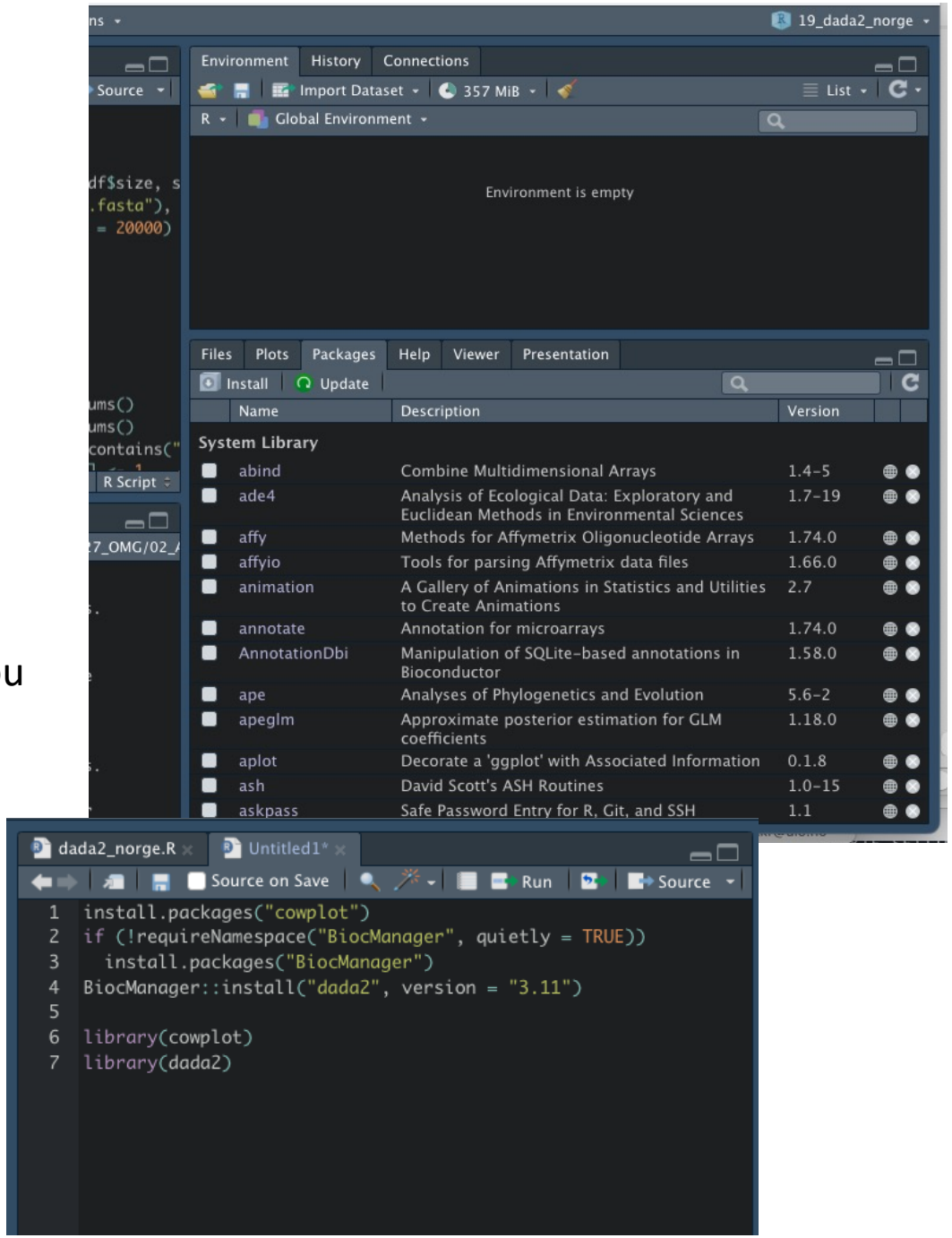


Loading packages

- Once a package is installed you need to load it into the environment for the functions to be available for the *current session*
- Point-and-click is available (but not recommended since you want to have the libraries in your script)
- The command for loading a package is library():

```
library(cowplot)
library(dada2)
```

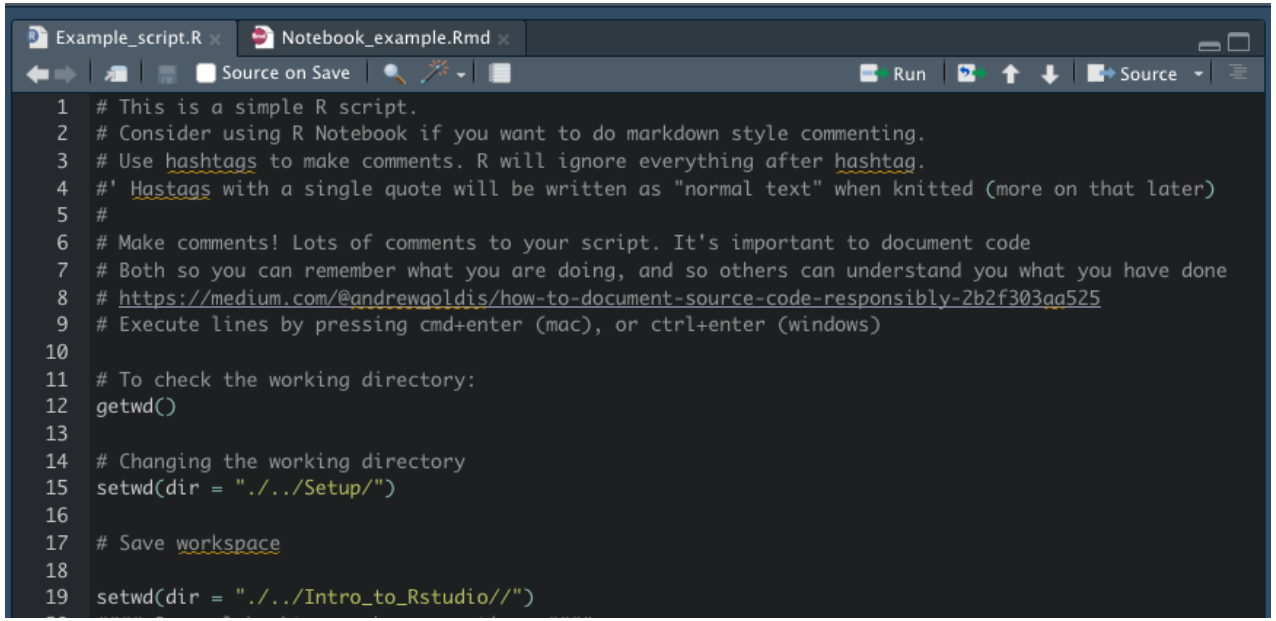
- When you close Rstudio the package will be unloaded and you have to reload the package when you start you next session



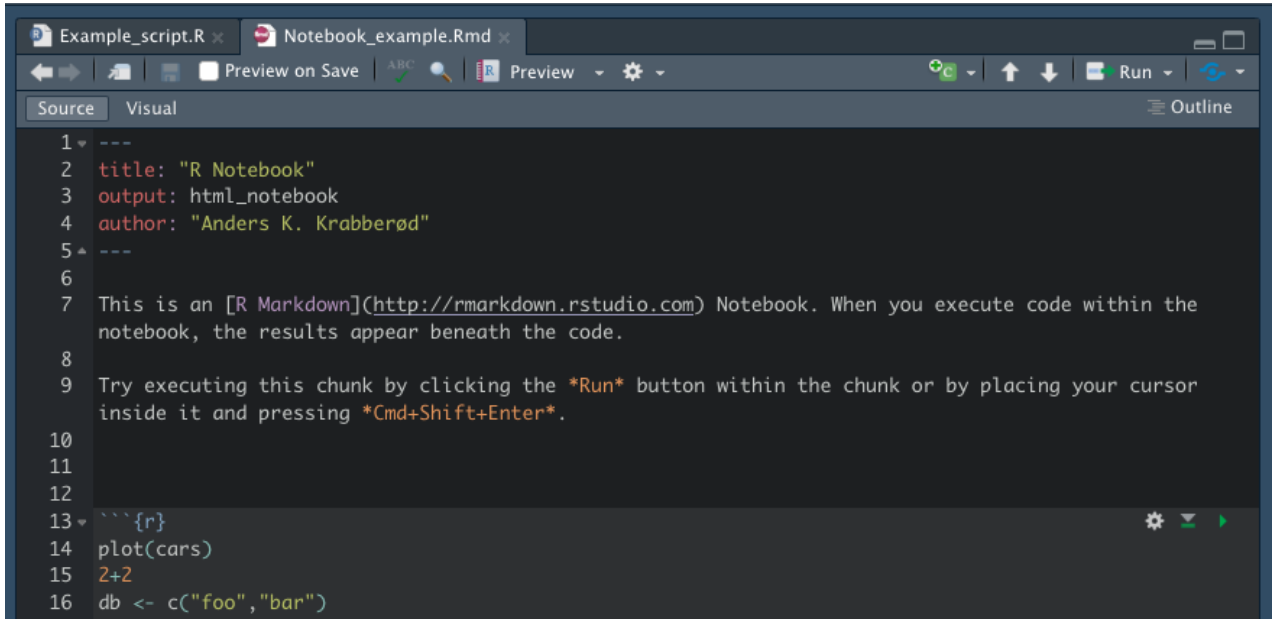
Use scripts or R Notebook

- Scripts and R notebooks allows you to save your process and makes it easy to share the work, or re-run any analysis.
- Scripts are simple text files that contains R commands executed line by line. All lines not preceded by a # will be executed
- R Notebooks are more complex, with chunks of code and running text together in the same document. Only the chunks will be executed (more on that later).

R script example:



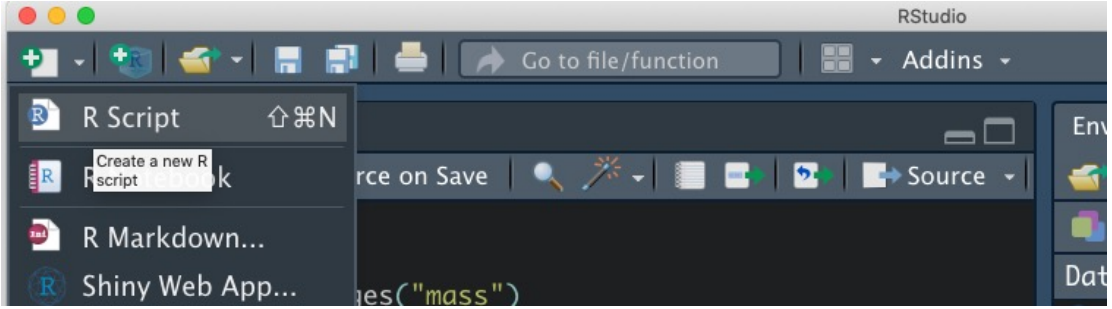
R Notebook example:



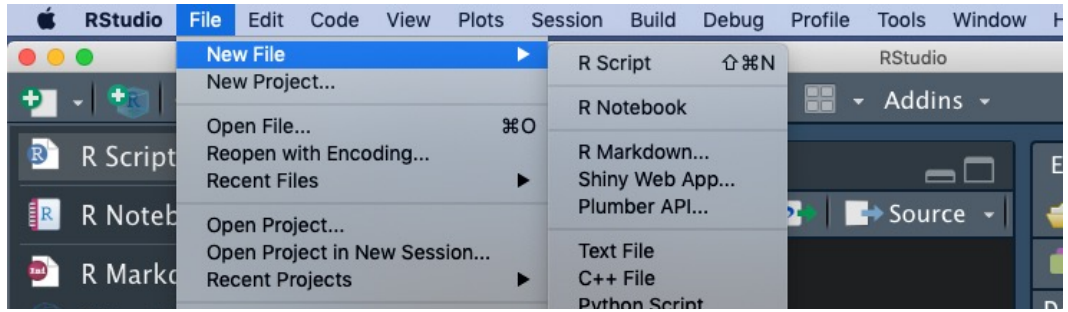
Create an R script

- Scripts are simple text files that contains R commands executed line by line.

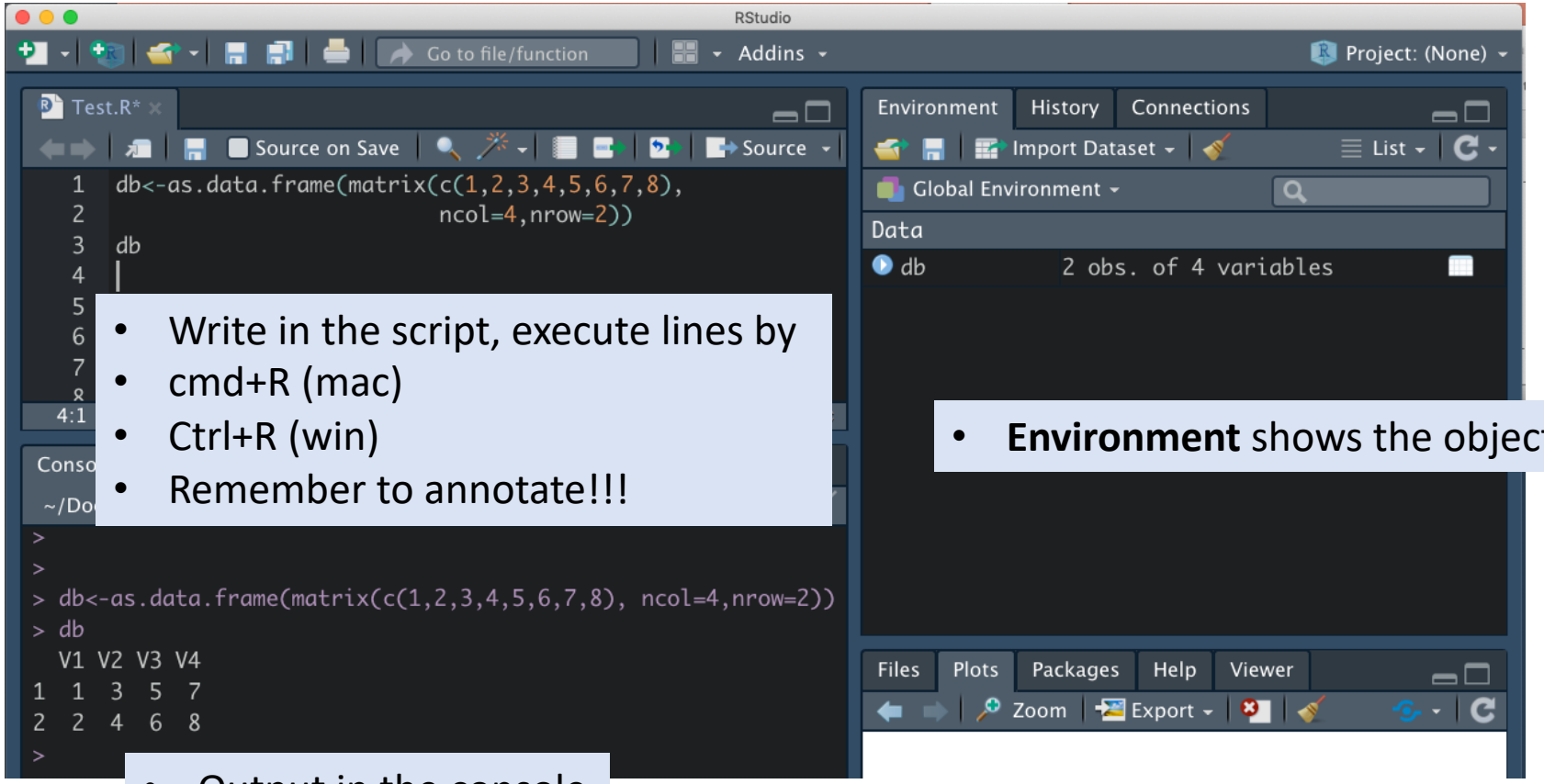
Click icon with a document and a + sign



OR click File -> New File -> R Script



Using Scripts



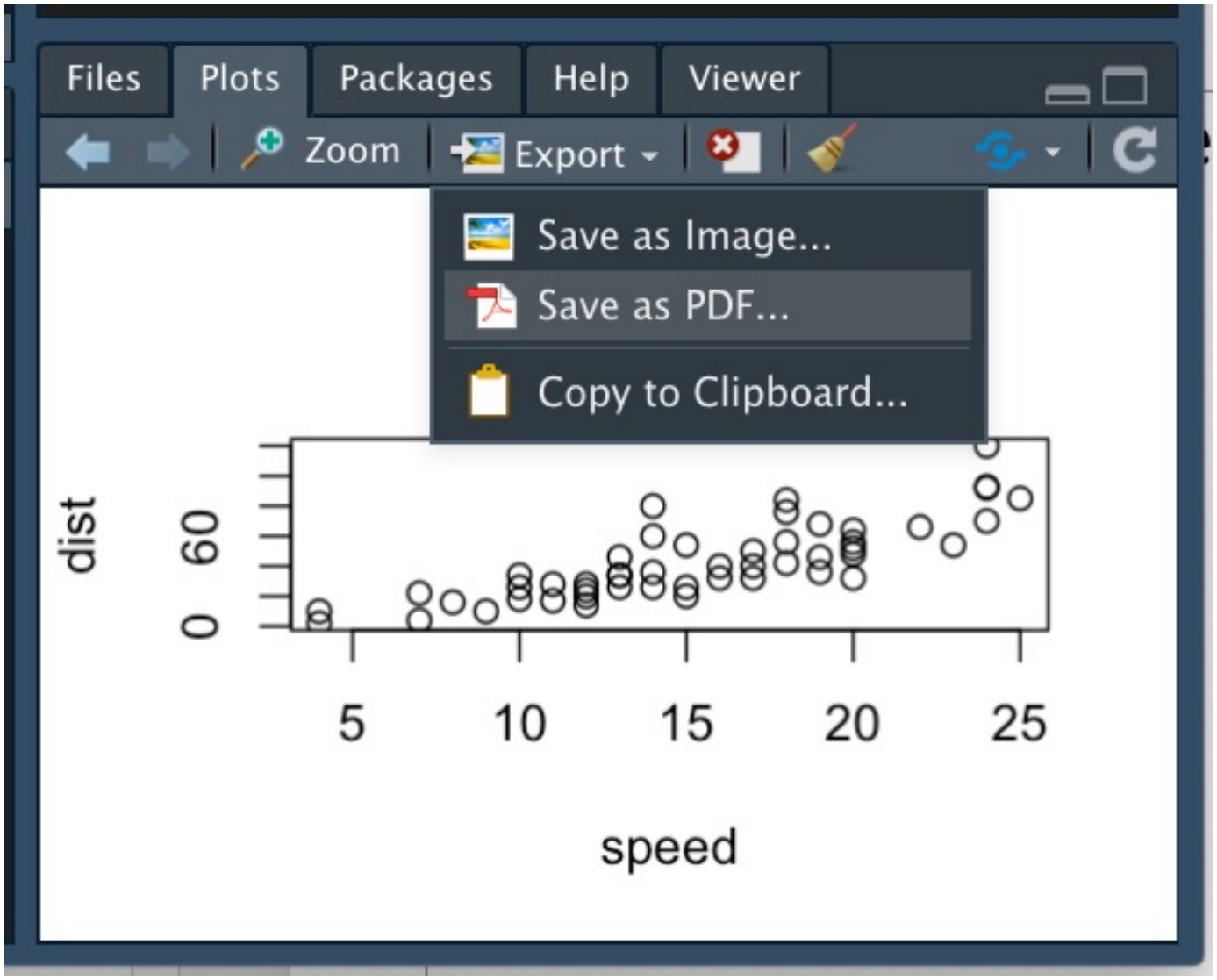
- Write in the script, execute lines by
- cmd+R (mac)
- Ctrl+R (win)
- Remember to annotate!!!

- **Environment** shows the object

- Output in the console

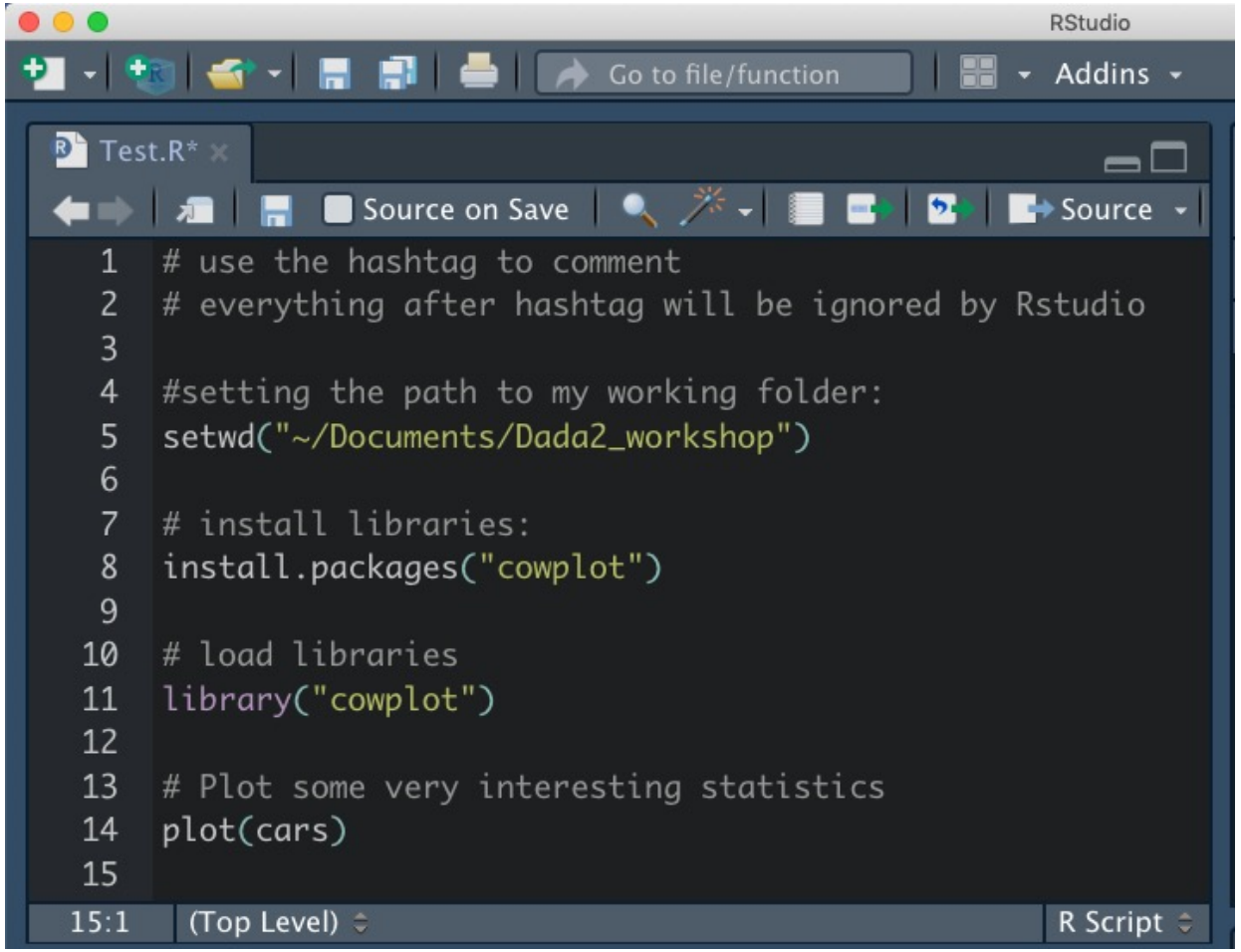
Plotting plots and other dots

- Plots will appear in the *plots* tab and can be exported in various formats.
- Additionally it is possible to give plotting commands in the script that will export the graphics to your working directory.
- The package *ggplot2* is very powerful and contains several plotting functions.



Comment and annotate your script!!!

- What the code does
- How the code does it
- How to use the code

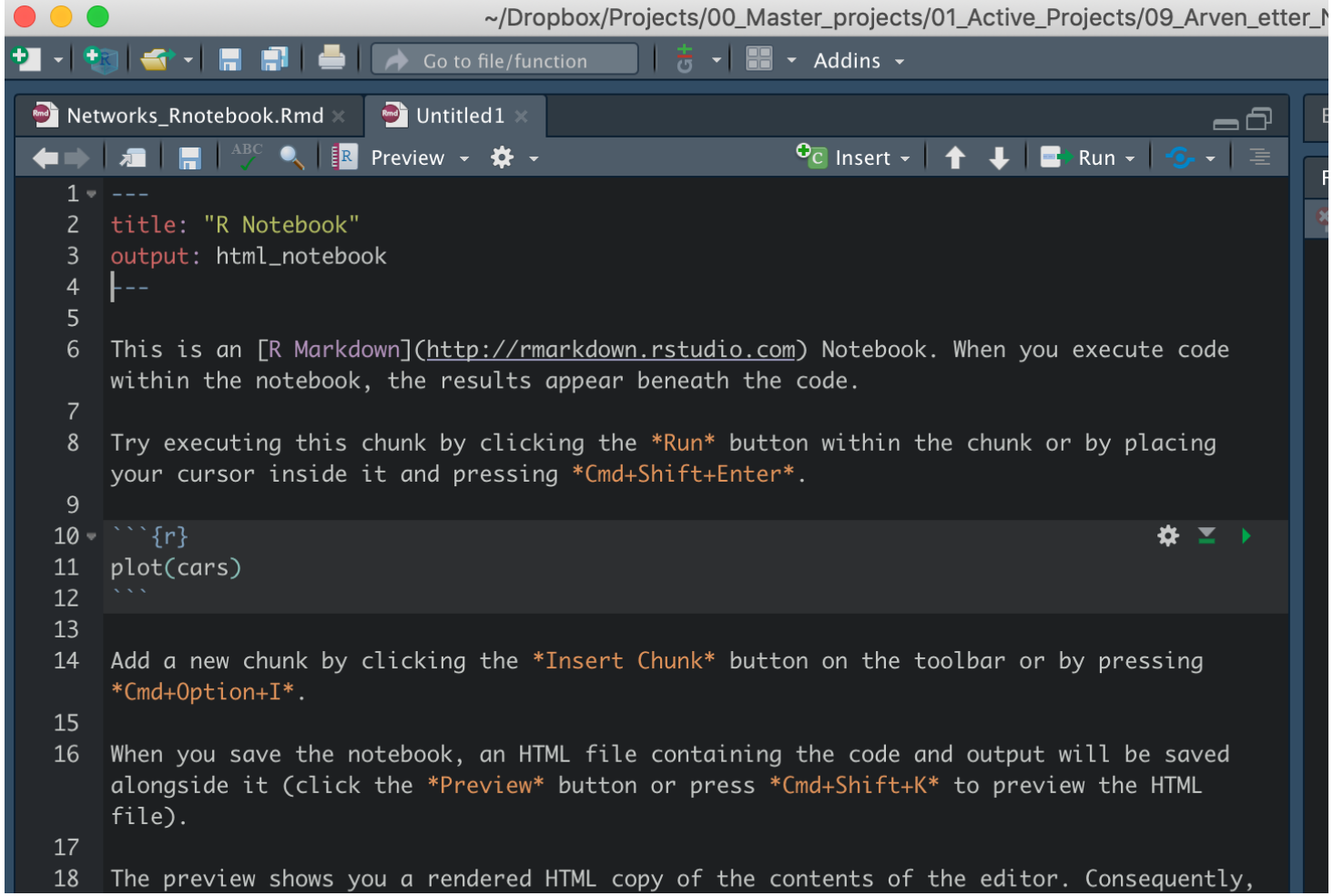


R notebooks and markdown

- R Notebook is an alternative to “simple” script in Rstudio.
- **Advantage:** easy to export in other easy-to-read formats (i.e. html, pdf, word, presentations).
- Markdown language is an easy way of formatting using plain text
- R Notebook is somewhat more powerful with additional options for formatting.
- Can run chunks of code from other languages *within* Rstudio
- **Disadvantage**: Not compatible with (standalone) R, which is often used on clusters and servers. For instance running R on Saga can be done with a simple script.

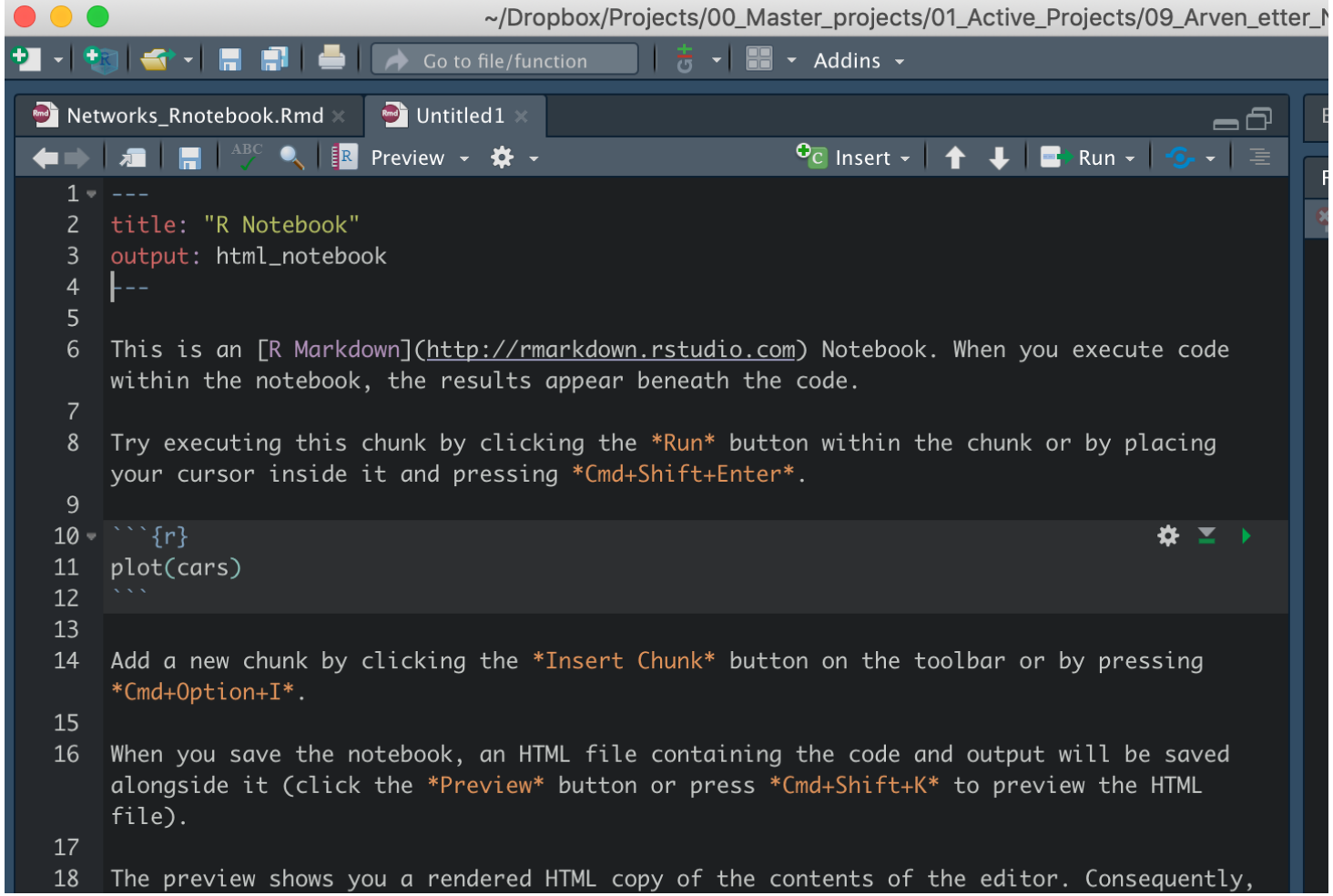
R Notebook

- The first lines are called the YAML (**Y**AML **A**in't **M**arkup **L**anguage') and contains the metadata and options for the entire document such as the author name, date, output format, etc. The beginning and the end of the YAML is marked by three dashes (---).
- Code is run in chunks placed bewteen three *opening single quotation marks*, in the curly bracket the type of code is defined. For R it is {r}.
- Inserting a chunk of code is done by pressing **Cmd+Option+I** on macs or **Ctrl+Alt+I** on Windows



R Notebook

- Text between the chunks will not be executed, and can be formatted with the markdown syntax.
- When you save the R Notebook a html file containing the output will be saved.
- The format can be changed to pdf, word etc.
- If you want to learn more about R Notebook and R Markdown this is a good source: https://intro2r.com/rmarkdown_r.html



Save and load your work

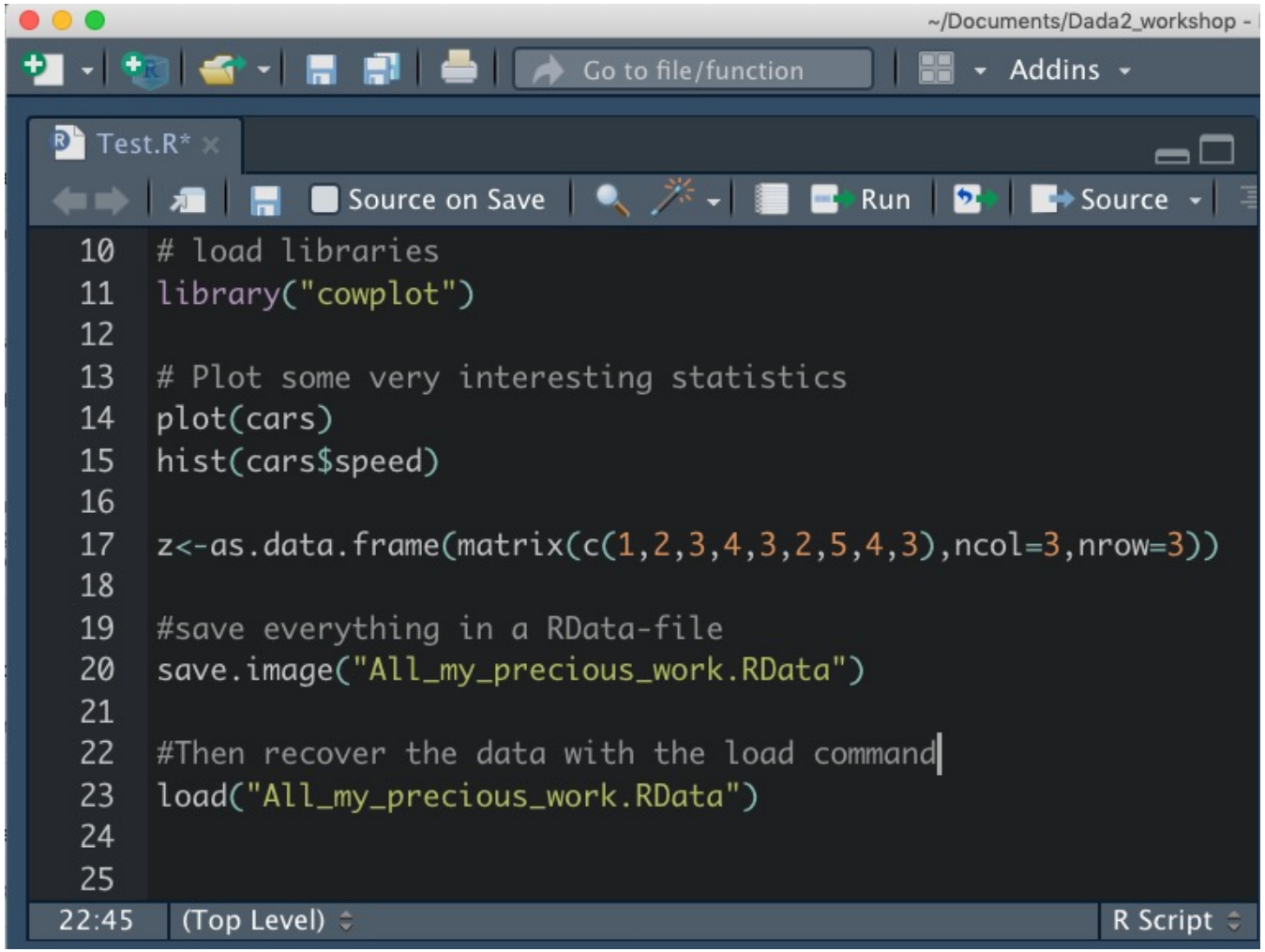
- You can save your “workspace” this will save all created objects and variables (but **not** plots and packages).

  `save.images(“filename.Rdata”)`

- Retrieve with

  `load(“filename.Rdata”)`

- Easy way to keep all analyses without having to rerun everything.
- You can also save single objects or a selection of objects with saveRDS()



Use R-projects

- This will set the default working directory for the particular project, and makes it easy to save everything in the same folder.
- Very helpful when working on several different projects
- Also very easy to integrate with *github* and version control with the option to push and pull repositories (not covered in this workshop)
- Or for sharing all data with somebody else using Rstudio