Getting Setup*

Chapter 0

Anna Ziff

R Workflow for Economists

Contents

*Please contact anna.ziff@duke.edu if there are errors.

About R

R is a programming language and work environment for data analysis. R is especially well-suited for data work and graphics, although it provides the base for many types of analyses. It is widely used in economics across sectors. The benefits of R are as follows.

- Free. Self-explanatory.
- Open source. Any user can develop functions and packages. These packages are freely available, resulting in the constant expansion and improvement of R and its capabilities.
- Flexible. R incorporates many of the strengths of the alternatives. It can be used to clean, analyze, and visualize data. With the aid of packages, it can also be used for optimization, web scrapping, network analysis, text analysis, machine learning, and many other tasks.
- Fast. R is designed for efficiency, especially for certain data structures. Even a novice user can write very efficient code.
- Relatively simple. Even though R has a steeper learning curve than STATA, it is much more straightforward than C.

Despite these benefits, there are a few downsides that any user must consider.

- Difficulty. Is the user committed to learning how to use R to maximize its benefits? Often, code needs to be read by those other than the author. Are these other individuals equipped to understand R syntax and semantics?[1]
- Errors. The user-written packages are heterogeneous in quality. Some may have poor documentation or even mistakes. Sometimes, authors do not update their code or respond to error reports, resulting in commands that do not work for more recent versions of R.

Installation

R

1. R is available for free from the Comprehensive R Archive Network (CRAN). Go to this website.
2. Click the download link that corresponds to your operating system.
    - Mac. Select the version that corresponds with the version of your operating system. Follow the installation instructions.
    - Windows. Click "Base" and download the installer. Follow the installation instructions.
    - Linux. Click on your distribution. Follow the installation instructions.

Confirm that installation was successful by opening the R program. When you open R, you will see a console that looks like Figure 1, or at least very similar. Every time you open R, a new session begins. At the start of each session, the console prints some information, like the version of R and some basic help commands. The line with `>` is the command line. This is where you can type commands. Try typing `print("Hello world!")` and press `Enter` or `Return`.

RStudio

While the R application is perfectly functional on its own, it is not as convenient as other applications. Specifically, we are interested in a more user-friendly Graphical User Interface (GUI). The GUI comprises the menus available to users for point-and-click operations. RStudio is a free software with an accessible GUI.

1. Navigate to the RStudio IDE webpage.

[1] The syntax of programming languages comprises the symbols, words, and structure of the code. The semantics of programming languages comprise the meaning behind the symbols, words, and structure.

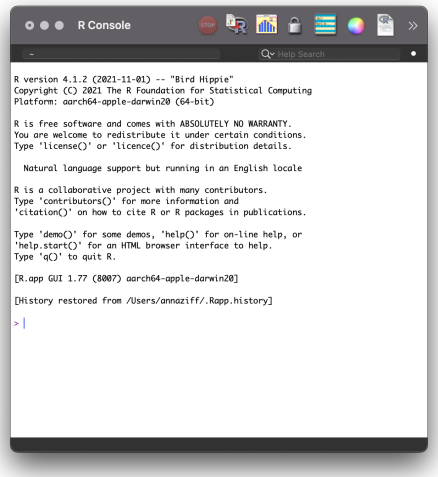

Figure 1: R Console

2. Choose the free version of RStudio Desktop.
3. Select your operating system and follow the installation instructions.

Confirm that the installation was successful by opening RStudio. The workspace should look very similar to Figure 2.

Using RStudio

There are three segments of the initial RStudio workspace: the console (left segment), which is the same as the basic R console, the workspace environment (upper right segment), and miscellaneous displays (lower right segment).

There is one more segment that is crucial: the script editor. Although you can type commands directly into the console, it is much easier and better practice to execute code from scripts. To open the script editor with an empty file, click `File > New File > R Script`. The file extension is `.R`. With all four segments of the RStudio workspace open (Figure 3), we can now go through each one in detail.

Console

Just like in the R application, you can type commands here directly at the angle bracket, `>`. Try typing `print("Hello world!")`. The output is printed in the same window. If you quit RStudio and reopen it, the commands you typed in the console will not be saved. For this reason, the console should be used to print output and test commands, not to write your entire analysis.

Script Editor

The script editor is where you can write, edit, and save your code. The code can be executed with the output printed to the console. In your blank R script, type the following lines.

```
# Title: R_00_Practice.R

greeting <- "Hello world!"
print(greeting)
```

Execute the code by highlighting all lines. Then click `Run`, circled in Figure 4.

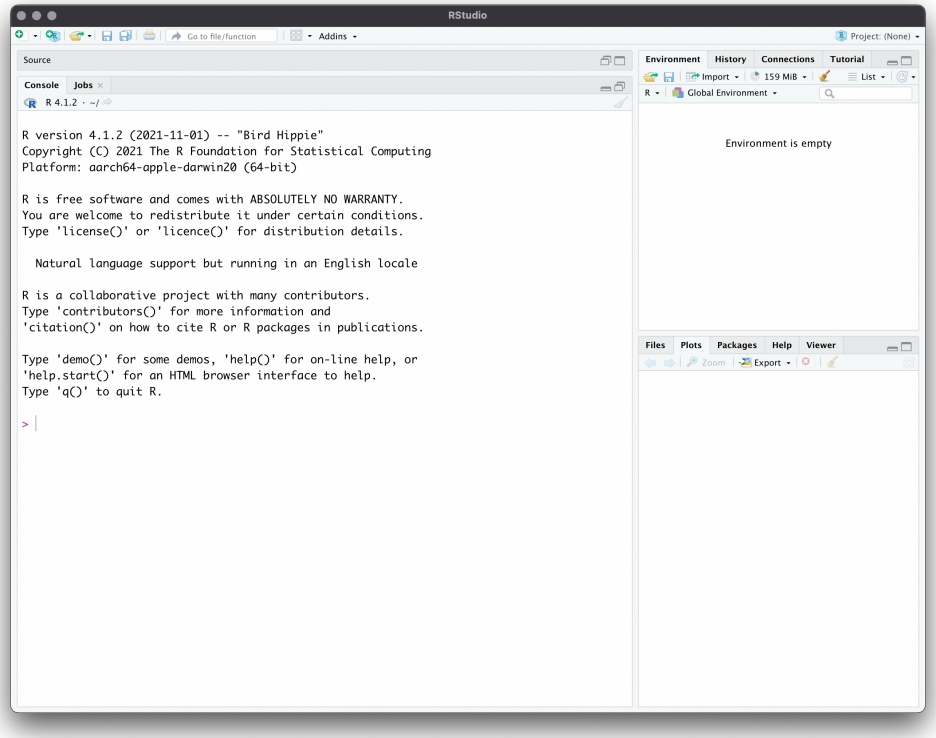

Figure 2: RStudio Opening Console

Workspace

Once you run your code, you will see a change in the workspace environment. Any object, structure, or function that is created in your code will be listed there. To remove an object, structure, or function, use the command `rm()`. It is good practice to clear the workspace before any code is executed. Add `rm(list = ls())` to the top of your code to clear everything from the workspace. Note that the command `ls()` lists all of the contents of the workspace.

```
# Title: R_00_Practice.R

rm(list = ls())

greeting <- "Hello world!"
print(greeting)
```

The `History` tab keeps track of recently used commands. The `Connections` tab is useful for SQL integration (outside the scope of this class). The `Tutorial` tab has instructions to access some tutorials.

Miscellaneous Displays

The lower right segment contains several miscellaneous displays, each of which are useful for different steps of the R workflow.

Files

The first display, `Files`, lists the folders of your computer. Clicking on the folders allows you to explore the file structure of your computer. Navigate to the folder in which you will save your code. At the top of

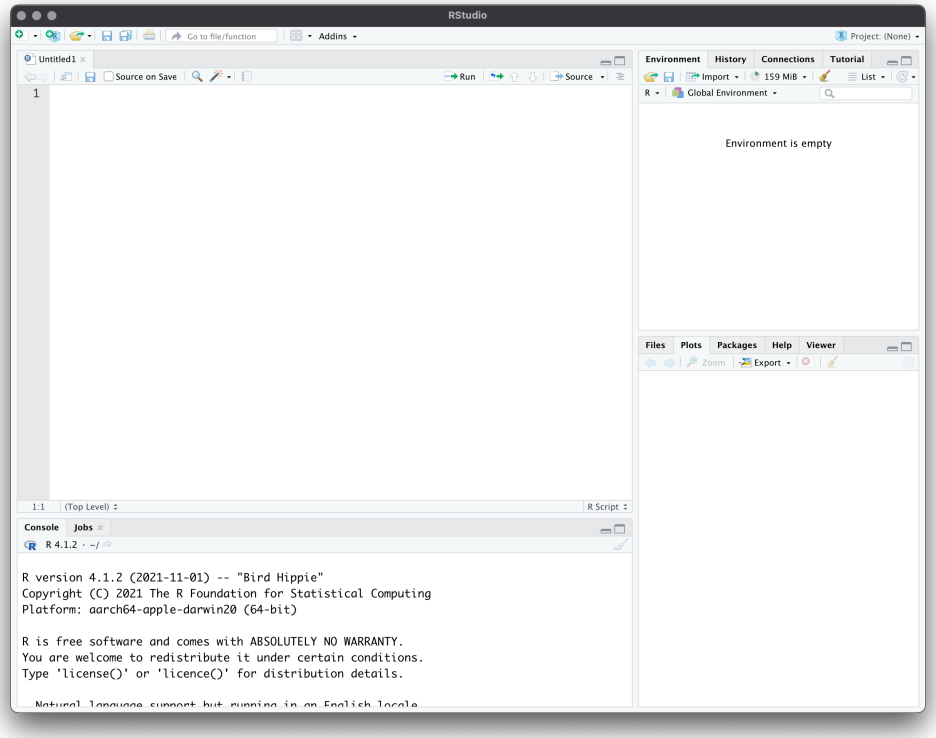

Figure 3: RStudio Full Workspace

the segment, you can see the file path to get there. This is useful when you need to change your working directory. In the console, experiment with the commands `getwd()` to print the current directory and `setwd()` to navigate the console to the `Practice-Exercises` directory. Add this to your R script. You will have a different filepath in the below code than the generic one.

```
# Title: R_00_Practice.R

rm(list = ls())
setwd("path/to/file")

greeting <- "Hello world!"
print(greeting)
```

Save the file with the name `R_00_Practice.R` to your current directory. This is done by clicking `File > Save`.

Plots

The `Plots` window is useful when you are creating graphs, as they show up directly. The graphics demo, typed directly into the console, showcases some examples of R's graphing capabilities.

```
demo(graphics)
```

Packages

The `Packages` tab displays all the external libraries on your computer. R is open source, meaning that anyone can publish functions to expand upon what is available for R users. The shareable code and documentation

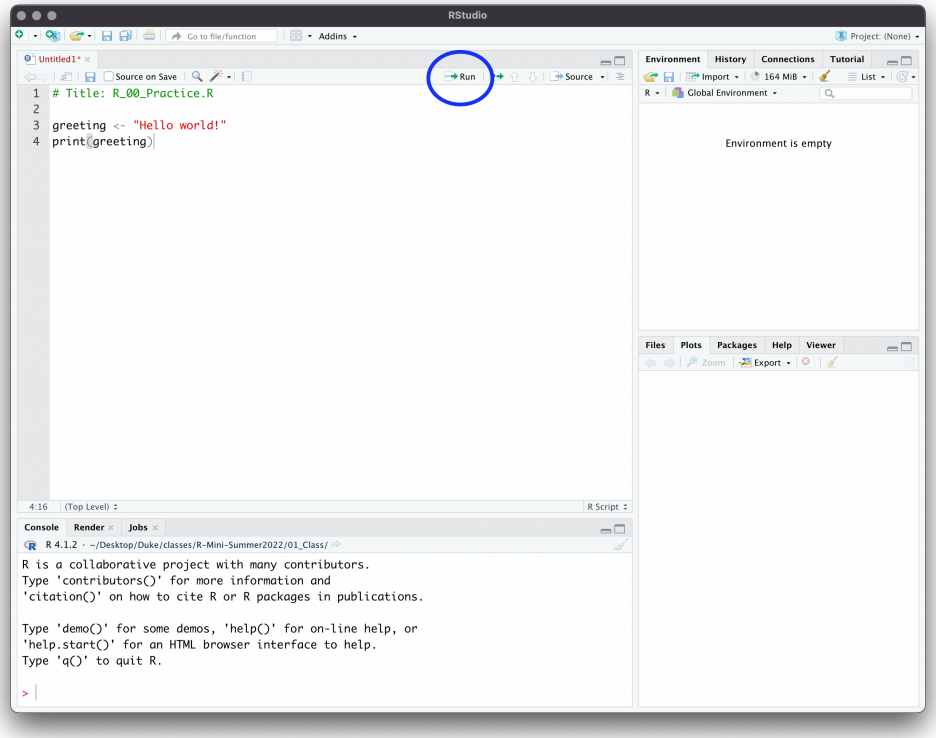

Figure 4: RStudio Run Button

is called a package or library. The advantage of packages is that there may already be an implementation of a problem you aim to solve. For example, if you want to implement a method from an econometrics paper, a package for that method may already exist. If the package is well done, it may be more efficient and accurate than your own implementation.

The Comprehensive R Archive Network (CRAN) is the official repository of R packages. Most of the packages you use will be published on CRAN. The function to install a package is `install.packages()`. Here is an example of how to install a package.

```
install.packages("ggplot2")
```

Once you install a package, the code and documentation are stored on your computer. You do not need to re-install the package every R session unless you want to ensure you have the most updated version of it. However, the package does need to be loaded every R session.

```
library(ggplot2)
```

If you want to specify which package the function comes from, you can use two colons, `::`. Using this format allows you to skip loading the package, although the package does need to be installed. You may see published code or answers on Stack Exchange that use this format. Here is an example. Sometimes, packages are only available on GitHub or you will want to use a more recent version that the package authors have not yet published to CRAN. To install these packages from GitHub, we need to access the `install_github()` function from the package `devtools`.

```
devtools::install_github("https://github.com/tidyverse/ggplot2")
```

It is common that different packages define functions that have the same name. If you call one of these functions, R will default to the package that was loaded later (type `search()` in the command line to see the

order in which packages were loaded). To notify you of this issue, R throws a warning message upon loading the package. The message lists the objects that are masked and the packages with the conflict.

```
install.packages("dplyr")
```

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

The double colon, `::` can be used to solve this issue. If you want to use one of these functions, you can specify which package R should reference. This is not always necessary, but it can help avoid errors if R is referencing a different package than what you expect.

```
base::union(1:10, 7:12)
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10 11 12
```

Generally, it is good practice to load the packages you will use at the top of your R script. Edit your R script to load `ggplot2`.

```
# Title: R_00_Practice.R

rm(list = ls())
setwd("~/Desktop/Duke/R/Assignments-aziff/Practice-Exercises")

library(ggplot2)

greeting <- "Hello world!"
print(greeting)
```

Help

The `Help` tab displays the documentation for functions. In the console, type the following.

```
help(print)
?print
```

You will see an explanation of the function, a description of the usage, a list of the arguments, other details, references, and examples. As you are learning, it is great practice to get accustomed to accessing the documentation of commands; and RStudio makes it easy! While you will need to search online for specific help and troubleshooting, the `?` command and the integrated `Help` window make the process of learning new commands as simple as possible.

Viewer

The `Viewer` tab is helpful when creating websites and applications that use R input and output (Shiny). This is outside the scope of the class.

Practice Exercises 0

1. The top of the script has a commented line: `# Title:  R_00_Practice.R`. The `#` signals to R not to execute the code. Add other lines to the header of your script for the author and date.
2. Access the documentation for `mean()` from the console. Familiarize yourself with the structure of the help documentation. Take note of the following sections: Description, Usage, Arguments, Value, Examples.
3. Several packages are combined in the collection `tidyverse`. These are all useful for data science, and we will be using functions from several of them in this class. Install this package. It takes a while!
4. The collection `tidyverse` contains `ggplot2`. Delete `library(ggplot2)` in `R_00_Practice.R` and load `tidyverse` instead.

Further Reading

This website goes through the most important features of the script editor. If you have never used RStudio before, skim through these features. Test out the keyboard shortcut to execute code.

The information above primarily comes from chapters 2 and 3.2-3.4 of Boehmke (2016).

References

Boehmke, Bradley C. 2016. *Data Wrangling with R*. Use R! Springer. https://link-springer-com.proxy.lib.duke.edu/content/pdf/10.1007%2F978-3-319-45599-0.pdf.