

Economics of Public and Social Issues
ECO 3253, Fall 2022

Jonathan Moreno-Medina

2022-08-17

Contents

Chapter 1

About

Blackboard

This is a *sample* book written in **Markdown**. You can use anything that Pandoc’s Markdown supports; for example, a math equation $a^2 + b^2 = c^2$.

1.1 Usage

Each **bookdown** chapter is an .Rmd file, and each .Rmd file can contain one (and only one) chapter. A chapter *must* start with a first-level heading: **# A good chapter**, and can contain one (and only one) first-level heading.

Use second-level and higher headings within chapters like: **## A short section** or **### An even shorter section**.

The `index.Rmd` file is required, and is also your first book chapter. It will be the homepage when you render the book.

1.2 Render book

You can render the HTML version of this example book without changing anything:

1. Find the **Build** pane in the RStudio IDE, and
2. Click on **Build Book**, then select your output format, or select “All formats” if you’d like to use multiple formats from the same book source files.

Or build the book from the R console:

```
bookdown::render_book()
```

To render this example to PDF as a `bookdown::pdf_book`, you'll need to install XeLaTeX. You are recommended to install TinyTeX (which includes XeLaTeX): <https://yihui.org/tinytex/>.

Chapter 2

Getting Started with Data in R

2.1 Why R again?

2.1.1 Why are we learning R? I wanted to learn about social issues...

This class *is* about social issues in economics. But what are those social issues?

- Economic mobility
- Effects of education
- Pollution and climate change
- and so on...

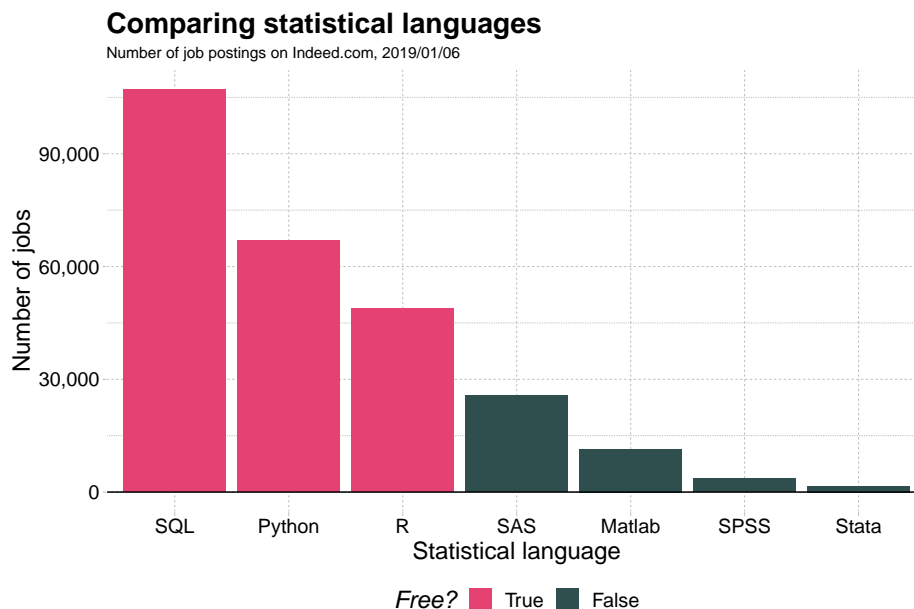
How can we know if going to school increases wages? Or if economic mobility is low or high?

We need to *analyze* data! We can do that analysis by hand... but that would be very time consuming. Or we can use a super calculator with amazing capabilities to explore data, maps, etc: enter **R** and **R Studio**.

2.1.2 Ok, but why R?

- **R** is free and open source!
- **R** has a vibrant online community!
- **R** is very flexible and powerful — adaptable to nearly any task (*e.g.*, correlations, econometrics, spatial data analysis, machine learning, web scraping, data cleaning, website building, teaching.)
- Employers like **R** over alternatives

2.1.3 Added benefits of learning R



2.2 Key concepts before we start:

Before we can start exploring data in R, there are some key concepts to understand first:

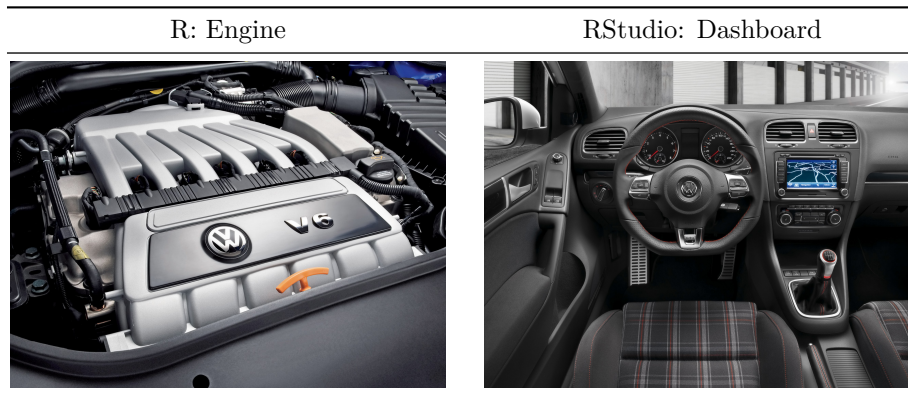
1. What are R and RStudio?
2. How do I code in R?
3. What are R packages?

We'll introduce these concepts in upcoming Sections ??-??. If you are already somewhat familiar with these concepts, feel free to skip to Section ?? where we'll introduce our first data set: all domestic flights departing a New York City airport in 2013. This is a dataset we will explore in depth in this book.

2.2.1 What are R and RStudio?

For much of this book, we will assume that you are using R via RStudio. First time users often confuse the two. At its simplest:

- R is like a car's engine.
- RStudio is like a car's dashboard.



More precisely, R is a programming language that runs computations while RStudio is an *integrated development environment (IDE)* that provides an interface by adding many convenient features and tools. So just as having access to a speedometer, rearview mirrors, and a navigation system makes driving much easier, using RStudio's interface makes using R much easier as well.



2.2.2 Installing R and RStudio

Note about RStudio Server or RStudio Cloud: If your instructor has provided you with a link and access to RStudio Server or RStudio Cloud, then you can skip this section. We do recommend after a few months of working on RStudio Server/Cloud that you return to these instructions to install this software on your own computer though. You will first need to download and install both R and RStudio (Desktop version) on your computer. It is important that you install R first and then install RStudio second.

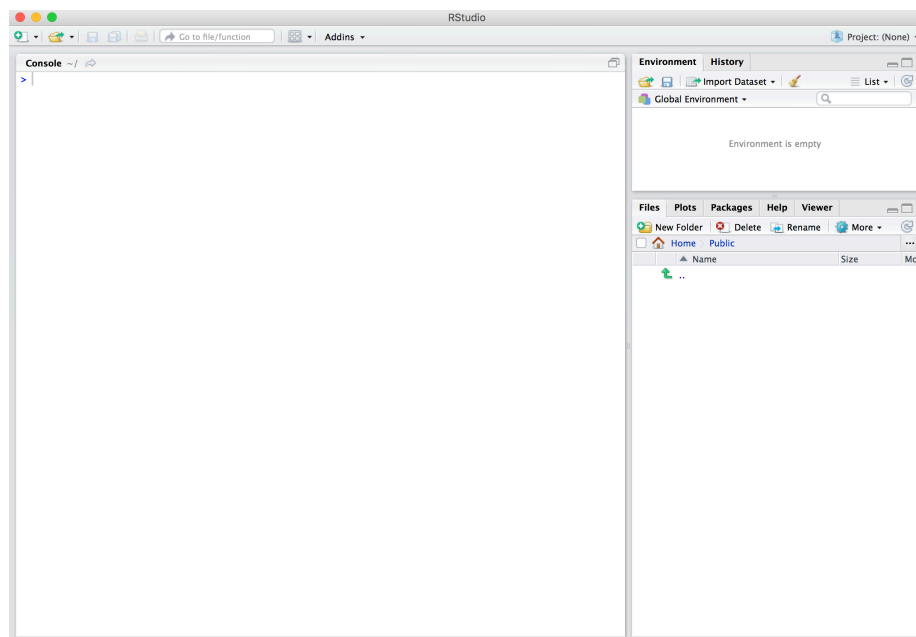
1. **You must do this first:** Download and install R.
 - If you are a Windows user: Click on “Download R for Windows”, then click on “base”, then click on the Download link.
 - If you are macOS user: Click on “Download R for (Mac) OS X”, then under “Latest release:” click on R-X.X.X.pkg, where R-X.X.X is the version number. For example, the latest version of R as of August 10, 2019 was R-3.6.1.
2. **You must do this second:** Download and install RStudio.
 - Scroll down to “Installers for Supported Platforms” near the bottom of the page.
 - Click on the download link corresponding to your computer's operating system.

2.2.3 Using R via RStudio

Recall our car analogy from above. Much as we don't drive a car by interacting directly with the engine but rather by interacting with elements on the car's dashboard, we won't be using R directly but rather we will use RStudio's interface. After you install R and RStudio on your computer, you'll have two new programs AKA applications you can open. We will always work in RStudio and not R. In other words:

R: Do not open this	RStudio: Open this
	

After you open RStudio, you should see the following:



Note the three panes, which are three panels dividing the screen: The *Console* pane, the *Files* pane, and the *Environment* pane. Over the course of this chapter, you'll come to learn what purpose each of these panes serve.

2.3 How do I code in R?

Now that you're set up with R and RStudio, you are probably asking yourself "OK. Now how do I use R?" The first thing to note is that unlike other statistical software programs like Excel, STATA, or SAS that provide point and click interfaces, R is an interpreted language, meaning you have to enter in R commands written in R code. In other words, you have to code/program in R. Note that we'll use the terms "coding" and "programming" interchangeably in this book.

While it is not required to be a seasoned coder/computer programmer to use R, there is still a set of basic programming concepts that R users need to understand. Consequently, while this book is not a book on programming, you will still learn just enough of these basic programming concepts needed to explore and analyze data effectively.

2.3.1 Basic programming concepts and terminology

We now introduce some basic programming concepts and terminology. Instead of asking you to learn all these concepts and terminology right now, we'll guide you so that you'll "learn by doing." Note that in this book we will always use a different font to distinguish regular text from `computer_code`. The best way to master these topics is, in our opinions, "learning by doing" and lots of repetition.

- **Basics:**
 - *Console*: Where you enter in commands.
 - *Running code*: The act of telling R to perform an action by giving it commands in the console.
 - *Objects*: Where values are saved in R. In order to do useful and interesting things in R, we will want to *assign* a name to an object. For example we could do the following assignments: `x <- 44 - 20` and `three <- 3`. This would allow us to run `x + three` which would return 27.
 - *Data types*: Integers, doubles/numerics, logicals, and characters.
- *Vectors*: A series of values. These are created using the `c()` function, where `c()` stands for "combine" or "concatenate." For example: `c(6, 11, 13, 31, 90, 92)`.
- *Factors*: *Categorical data* are represented in R as factors.
- *Data frames*: Data frames are like rectangular spreadsheets: they are representations of datasets in R where the rows correspond to *observations* and the columns correspond to *variables* that describe the observations. We'll cover data frames later in Section ??.
- *Conditionals*:
 - Testing for equality in R using `==` (and not `=` which is typically used for assignment). Ex: `2 + 1 == 3` compares `2 + 1` to 3 and is correct R code, while `2 + 1 = 3` will return an error.
 - Boolean algebra: `TRUE/FALSE` statements and mathematical opera-

tors such as `<` (less than), `<=` (less than or equal), and `!=` (not equal to).

- Logical operators: `&` representing “and” as well as `|` representing “or.” Ex: `(2 + 1 == 3) & (2 + 1 == 4)` returns `FALSE` since both clauses are not `TRUE` (only the first clause is `TRUE`). On the other hand, `(2 + 1 == 3) | (2 + 1 == 4)` returns `TRUE` since at least one of the two clauses is `TRUE`.
- *Functions*, also called *commands*: Functions perform tasks in R. They take in inputs called *arguments* and return outputs. You can either manually specify a function’s arguments or use the function’s *default values*.

This list is by no means an exhaustive list of all the programming concepts and terminology needed to become a savvy R user; such a list would be so large it wouldn’t be very useful, especially for novices. Rather, we feel this is a minimally viable list of programming concepts and terminology you need to know before getting started. I’m confident you can learn the rest as you go. Remember that your mastery of all of these concepts and terminology will build as you practice more and more.¹

2.4 In Class Exercise

Try running the following commands in the console. What do you see for each one?

```
2*3
2*pi
log(10)
exp(2)
sqrt(25)
3==3
3==4
3<=4
3!=4
x <- c(1,3,2,5)# this is called a 'vector'
x #what do you see?
x <- c(1,6,2)
x #now what do you see?
y <- c(1,4,3) # USE ARROW!
length(x)
length(y)
x+y
#write this
write this again
```

¹If you truly insist on getting more information, you can check this link explaining some of the basics: <https://rstudio-education.github.io/hopr/basics.html> ;but again, **this is not required or expected**.

2.4.1 Errors, warnings, and messages

One thing that intimidates new R and RStudio users is how it reports *errors*, *warnings*, and *messages*. R reports errors, warnings, and messages in a glaring red font, which makes it seem like it is scolding you. However, seeing red text in the console is not always bad.

R will show red text in the console pane in three different situations:

- **Errors:** When the red text is a legitimate error, it will be prefaced with “Error in...” and try to explain what went wrong. Generally when there’s an error, the code will not run. For example, we’ll see in Subsection ?? if you see `Error in ggplot(...) : could not find function "ggplot"`, it means that the `ggplot()` function is not accessible because the package that contains the function (`ggplot2`) was not loaded with `library(ggplot2)`. Thus you cannot use the `ggplot()` function without the `ggplot2` package being loaded first.
- **Warnings:** When the red text is a warning, it will be prefaced with “Warning:” and R will try to explain why there’s a warning. Generally your code will still work, but with some caveats. For example, you will see in Chapter ?? if you create a scatterplot based on a dataset where one of the values is missing, you will see this warning: `Warning: Removed 1 rows containing missing values (geom_point)`. R will still produce the scatterplot with all the remaining values, but it is warning you that one of the points isn’t there.
- **Messages:** When the red text doesn’t start with either “Error” or “Warning”, it’s *just a friendly message*. You’ll see these messages when you load *R packages* in the upcoming Subsection ?? or when you read data saved in spreadsheet files with the `read_csv()` function as you’ll see in Chapter ?. These are helpful diagnostic messages and they don’t stop your code from working. Additionally, you’ll see these messages when you install packages too using `install.packages()`.

Remember, when you see red text in the console, *don’t panic*. It doesn’t necessarily mean anything is wrong. Rather:

- If the text starts with “Error”, figure out what’s causing it. Think of errors as a red traffic light: something is wrong!
- If the text starts with “Warning”, figure out if it’s something to worry about. For instance, if you get a warning about missing values in a scatterplot and you know there are missing values, you’re fine. If that’s surprising, look at your data and see what’s missing. Think of warnings as a yellow traffic light: everything is working fine, but watch out/pay attention.
- Otherwise the text is just a message. Read it, wave back at R, and thank it for talking to you. Think of messages as a green traffic light: everything is working fine.

2.4.2 Tips on learning to code

Learning to code/program is very much like learning a foreign language, it can be very daunting and frustrating at first. Such frustrations are very common and it is very normal to feel discouraged as you learn. However just as with learning a foreign language, if you put in the effort and are not afraid to make mistakes, anybody can learn.




Here are a few useful tips to keep in mind as you learn to program:

- **Remember that computers are not actually that smart:** You may think your computer or smartphone are “smart,” but really people spent a lot of time and energy designing them to appear “smart.” Rather you have to tell a computer everything it needs to do. Furthermore the instructions you give your computer can’t have any mistakes in them, nor can they be ambiguous in any way.
- **Take the “copy, paste, and tweak” approach:** Especially when learning your first programming language, it is often much easier to taking existing code that you know works and modify it to suit your ends, rather than trying to write new code from scratch. We call this the *copy, paste, and tweak* approach. So early on, we suggest not trying to write code from memory, but rather take existing examples we have provided you, then copy, paste, and tweak them to suit your goals. Don’t be afraid to play around!
- **The best way to learn to code is by doing:** Rather than learning to code for its own sake, we feel that learning to code goes much smoother when you have a goal in mind or when you are working on a particular project, like analyzing data that you are interested in.
- **Practice is key:** Just as the only method to improving your foreign language skills is through practice, practice, and practice; so also the only method to improving your coding is through practice, practice, and practice. Don’t worry however; we’ll give you plenty of opportunities to do so!

2.5 What are R packages?

Another point of confusion with many new R users is the idea of an R package. R packages extend the functionality of R by providing additional functions, data, and documentation. They are written by a world-wide community of R users and can be downloaded for free from the internet. For example, among the many packages we will use in this book are the `ggplot2` package for data visualization in Chapter ??, the `dplyr` package for data wrangling in Chapter ??, and the `moderndive` package that accompanies this book.

A good analogy for R packages is they are like apps you can download onto a mobile phone:

R: A new phone	R Packages: Apps you can download
	 

So R is like a new mobile phone: while it has a certain amount of features when you use it for the first time, it doesn't have everything. R packages are like the apps you can download onto your phone from Apple's App Store or Android's Google Play.

Let's continue this analogy by considering the Instagram app for editing and sharing pictures. Say you have purchased a new phone and you would like to share a recent photo you have taken on Instagram. You need to:

1. *Install the app*: Since your phone is new and does not include the Instagram app, you need to download the app from either the App Store or Google Play. You do this once and you're set. You might do this again in the future any time there is an update to the app.
2. *Open the app*: After you've installed Instagram, you need to open the app.

Once Instagram is open on your phone, you can then proceed to share your photo with your friends and family. The process is very similar for using an R package. You need to:

1. *Install the package*: This is like installing an app on your phone. Most packages are not installed by default when you install R and RStudio. Thus if you want to use a package for the first time, you need to install it first. Once you've installed a package, you likely won't install it again unless you want to update it to a newer version.
2. *"Load" the package*: "Loading" a package is like opening an app on your phone. Packages are not "loaded" by default when you start RStudio on your computer; you need to "load" each package you want to use every time you start RStudio.

Let's now show you how to perform these two steps for the `ggplot2` package for data visualization.

2.5.1 Package installation

There are two ways to install an R package. For example, to install the `ggplot2` package: