An R Guide

Joshua Allen

2023-07-18

Table of contents

Who is This For?

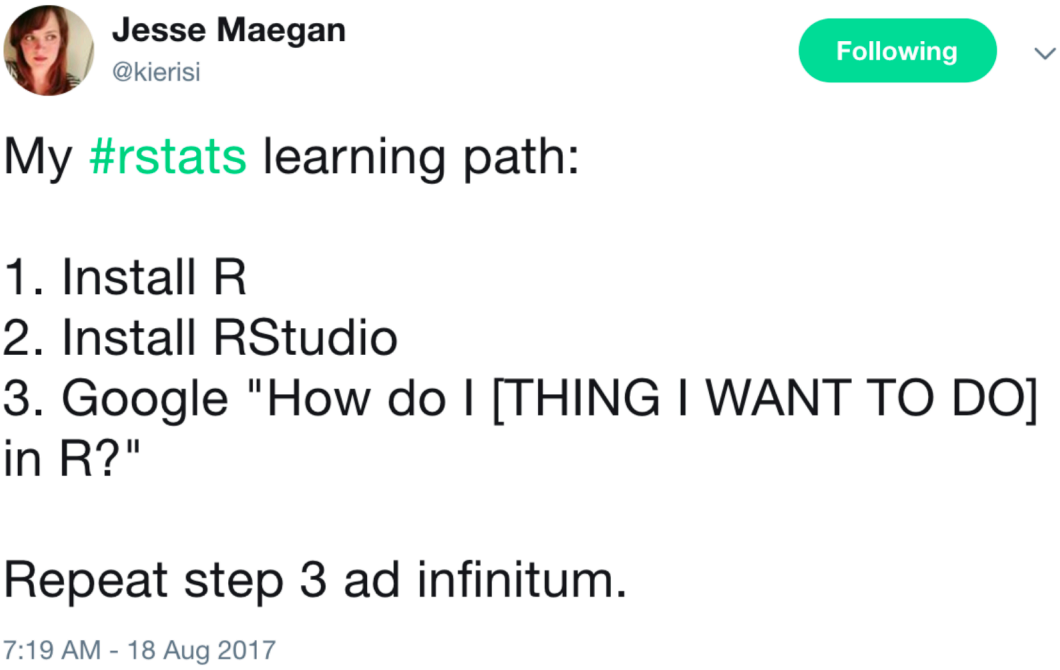
This guide was initially compiled for my Introduction to Political Science Research class however this book is really for anybody who is looking to get started in R and R Studio. The guide itself was to put it mildly a little chaotic and somewhat difficult to navigate. So this is my attempt to correct that. The goal of this guide is to provide a summary of the things that helped me and add some of my own flare. By no means is this meant to be a definitive guide to `R`. Hopefully it is just a gentle guide and a potential reminder of things you already know, or an introduction to things you did not know that R could do.

This "book" requires a significant amount of attribution to a ton of people. In particular Much of the content is based on https://github.com/uo-ec607/lectures by Dr. Grant McDermott, https://talks.andrewheiss.com/2021-seacen/01-tidyverse.html by Dr. Andrew Heiss as well as just general knowledge dispensed from his blog and peppering him with questions, and Graphic Design with ggplot2 by Dr. Cédric Scherer. I highly recommend that you check out these sources! As it stands right now I need to go back through and add attribution to the individual chapters. Most of this "book" is me just translating `R` into how I think and what helped me learn `R`.

Pep talk

> **!** Important
>
> "There is no way of knowing nothing about a subject to knowing something about a subject without going through a period of much frustration and suckiness… Push through. You'll suck less"
>
> - Hadley Wickham, author of ggplot2

I sucked at this initially. We all sucked at this initially. You will often miss a comma somewhere, and your code will not run. You will be puzzled why something is not working and get frustrated and not know why your code is running and why somebody else's code is running to find out later you misspelled something. I generated a ton of errors just compiling this guide. You eventually get better at figuring out the likely culprits and more carefully looking at your code. This is just something that comes with practice.



It is 100% okay to Google stuff when working on your analysis, problem sets, . We all do it! If I run into an issue, I do not know how to fix I immediately google it. Remember, the goal of this class is to teach you how to do research `R` is a minor part in the grand scheme of things when you are doing research. My goal is to teach you how to conduct research, not to make you a computer scientist or `R` developer.

:::

Part I

Installation and Basics

1 Before We Get to The Fun Stuff

One of the things that I have found is unintuitive for a lot of people when they are first learning `R` is how a computer thinks. A computer is very good at doing stuff quickly, but needs proper instructions. For things like an excel spreadsheet you can just open it and do things like `=Average(C2)` and it will just work. For Word you can just like drag and drop things without ever having to think about how the computer knows where an image is. `R` and other programming languages require some thought about things like this. So this chapter will introduce you to some of these things.

1.1 Working Directories

What is a working directory? Working directories are essentially the internal gps of the computer. You know(generally) where the stuff on your computer is you navigate by clicking on stuff and voila you found your stuff. Underneath the hood your computer is finding stuff by navigating along the working directory. On my computer to get to this R guide I click on my teaching folder, then I click on the Administrative Stuff folder, and then I click on the `R-guide.qmd` or `R-guide.html` file. This is what that process looks like for the computer

```
getwd() # this is the working directory for the r guide
```

```
[1] "/Users/josh/Library/CloudStorage/Dropbox/pols-3800"
```

Each `/` is essentially a click until you get to the final destination.

Heuristically you can think of it like this.

- You **can** put a box inside a box.
  - This means folders can contain other folders.
  - When you put a folder inside another folder this is called creating a subdirectory
- You **can** put a cat inside a box
  - This means you can put a file inside a folder.
- You **can** put a cat inside a box inside of a box
  - This means you can put a file inside a folder inside another folder
- You **cannot** put a box inside a cat
  - You cannot put a folder inside a document(thats not how computers work)
- You **cannot** put cat in a cat
  - This means you cannot put files inside other files. Admittedly this is where the metaphor breaks down.

So to include the cat pictures in this guide, I stored it in a folder called `figs` so to tell `R` to navigate to the picture, I do something like this.

```
# how you navigate to folders on mac
![](figs/cats-boxes.png) # this one way to include pics in a markdown doc
```

The `/` is you telling the computer that within the figs folder, there is a file I want you to include named `cats-boxes.png`. Windows does this slightly differently by doing something like

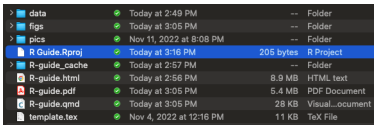
```
![](figs\cats-boxes.png)
```

To manually set your working directory you do this

```
setwd("path/to/your/files") # mac and linux
setwd("path\to\your\files") # windows
```

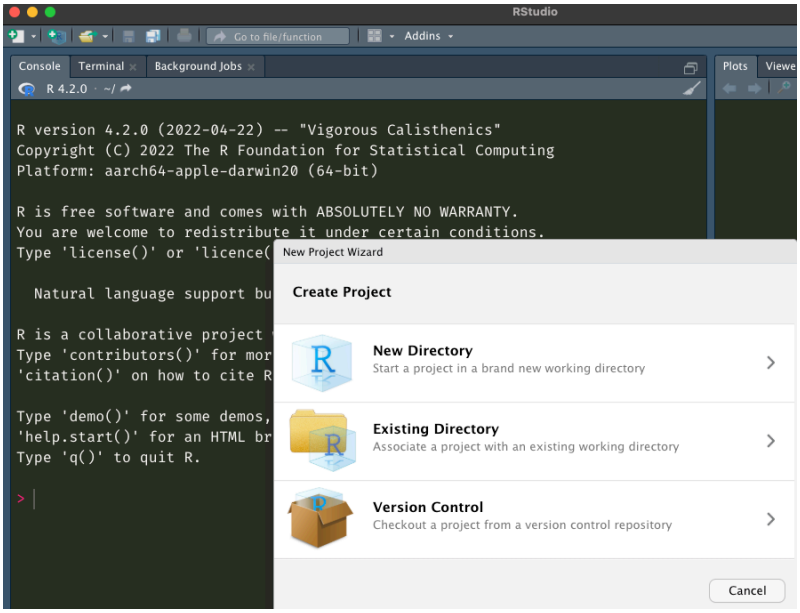
1.2 Working in Projects

However, working in R projects is best practice for various reasons. A projected-oriented workflow starts you off with a clean slate, so problem set 0 stuff is not still loaded. It also prevents you from constantly having to set the working directory.

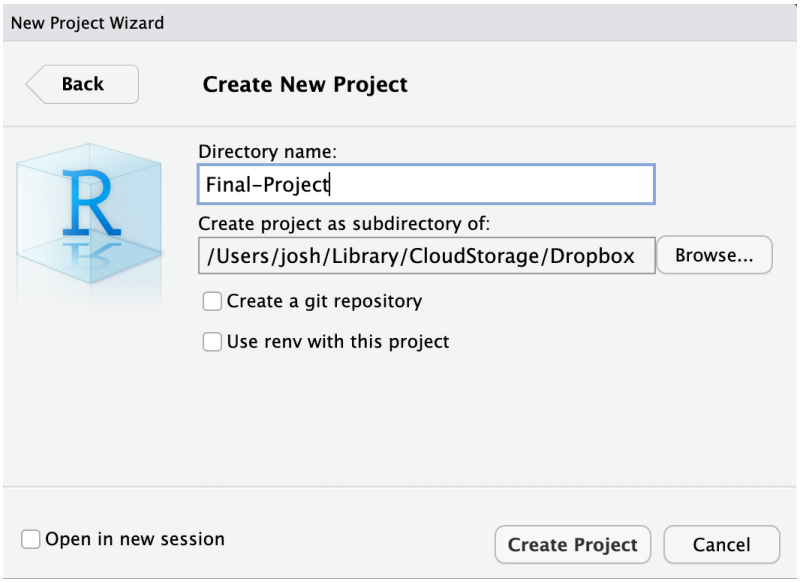


When you start working on a problem set click on the `.Rproj` file. This will open up RStudio, and you should be ready to go!

To open a new project open R-Studio. In the top left you will see something that looks like a blank page. If you look to the right there is an R with a + and like a weirdish blue background. Click on that and you should see a menu that looks like.



For your final projects you are going to want to click new directory, then click new project. You should be looking at a menu that looks like this



In the directory name portion put in an informative name like `Final-Project` or `ggplot lab`. If you look at the bottom of the box you will notice that it will place the folder in my Drop-

box folder. If you click on browse you can can change where the folder is put on your computer.

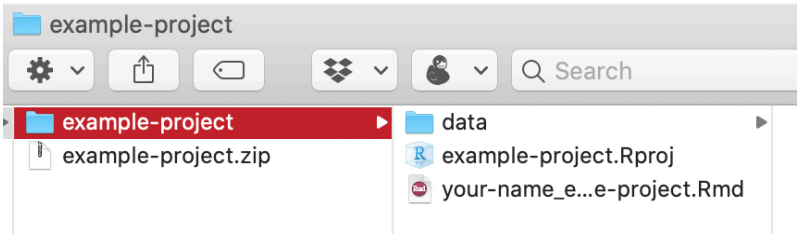
I strongly recommend you create a dedicated folder for your porjects instead of keeping everything on your desktop or downloads folder

1.3 Zip Files

To keep everything together in the same working directory I distribute the problem sets as Zip files. You will likely encounter these at some point in your professional life. If you want to send a folder via email to your boss or your direct reports that has lots of stuff like images than the chances are that you will have to send a Zip file because Outlook has a max file size of 33mb which is next to nothing. So it is worth spending a little time on how to unzip a file.

1.3.1 Mac

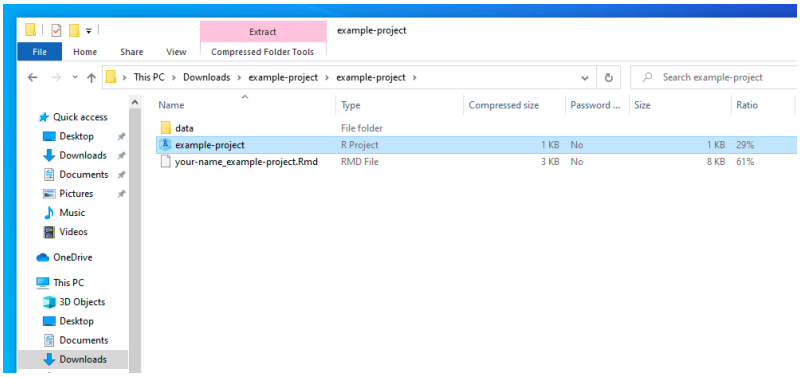
Fortunately for Mac users this is really easy. When you click download you will see something that looks like this



Double click on the downloaded .zip file. macOS will automatically create a new folder with the same name as the .zip file, and all the file's contents will be inside. Double click on the RStudio Project file (.Rproj) to get started.

1.3.2 Windows

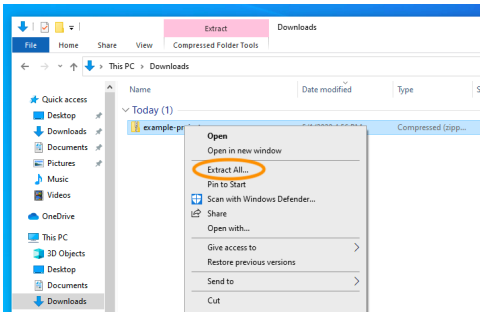
For Windows users this process is a bit more involved for reasons that are unclear to me. This can be at best an inconvenience at worst it can result in you doing all your work trying to save it and it disappearing into the abyss. One of the things that makes this incredibly confusing is that when you download the zip file it looks like a regular folder.



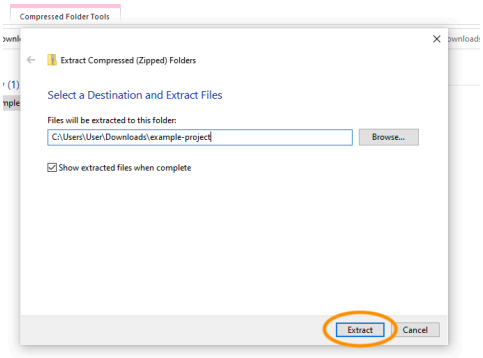
You can click around in it and even edit files within the zip file! Here's what it looks like—the only clues that this folder is really a `.zip` file are that there's a "Compressed Folder Tools" tab at the top, and there's a "Ratio" column that shows how much each file is compressed. All your hard work will be gone because it is saved in some temp directory that will take ages to find and likely not work all that well.

You most likely won't be able to open any data files or save anything, which will be frustrating.

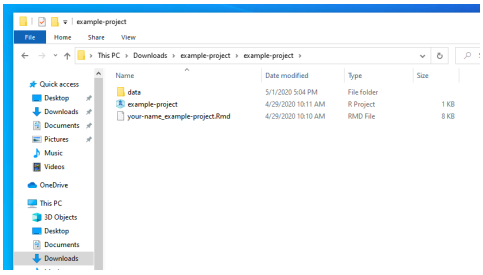
Instead, you need to right click on the `.zip` file and select "Extract All...":



Then choose where you want to unzip all the files and click on "Extract"



You should then finally have a real folder with all the contents of the zipped file. Open the R Project file and RStudio will point to the correct working directory and everything will work.



2 Installing R

> **!** Important Note Regarding Chromebooks
>
> If you have a Chromebook rather than a Windows or Mac computer, these installation instructions will not work. While there are ways to install R, RStudio, and other software on Chromebooks the process for doing so is different depending on what version of ChromeOS you have and may not be supported on older models. If this problem applies to you, please make an appointment or come by during office hours as soon as possible so we can figure out what the appropriate steps to get everything up and running are.

The easiest way to install `R` is to go to this website and select the operating system you use.

For Mac users it is important to select the correct download for your chip. If you are unsure click on the Apple logo in the top left of your screen and then click about this Mac.

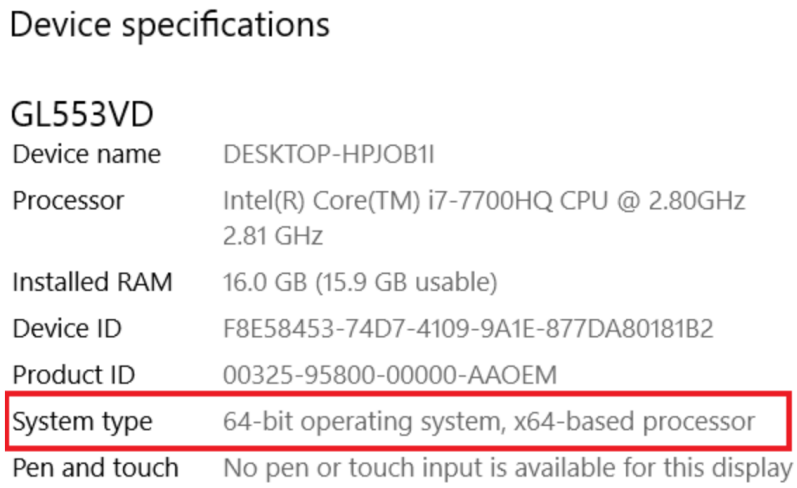
I would download the one that says arm64. If your chip says "intel something or other" then you would click on the one that says Intel Macs.

> **!** XQuartz for MacOS Users
>
> If you downloaded either of the MacOS versions above, you also need to download and install XQuartz. **This step is only for MacOS users**.

For Windows please click on this link and then follow this link and download the appropriate version of the software for your operating system. To check whether you need the X64 version

Figure 2.1: Look out for what chip your machine uses

or the X86 version press the windows + x keys and then click on system. You are looking for something that looks like this



Then choose 64 or 86.

2.1 R Tools and XCode

Occasionally some of the packages we will use may not be available for your version of R. This is not a problem since many package writers have the code online and we need to download them. This does require we have some additional things on our computer.

2.1.1 Windows

> ⚠ Required Step
>
> For some of the packages we will need we may need to have some additional things installed in order for it to work.

If you are on a Windows operating system, you will need to install Rtools 4.3 which can download by clicking on the hyperlink below. Rtools is required to install packages from source and, more importantly, anything that requires a C++ compiler.

- Download Rtools 4.3

Once again, during the installation process you can just leave things at their default settings, especially in the case of the installation directory since changing its default locations may result in errors during package compilation. After you have completed this step, you will need to install the required packages by running the code below in order to successfully render the Quarto document for problem set 0. **If you encounter errors when rendering the document, please ensure you have these packages installed.**

2.1.2 MacOS

> ⚠ Required Step
>
> For some of the packages we will need we may need to have some additional things installed in order for it to work. This is a required step for much of the code we will use throughout this course.

If you are on a MacOS operating system, you will need to install the Xcode developer tools. You can obtain the full MacOS development environment from the Apple AppStore using the link below. Xcode is required to install packages from source and, more importantly, anything that requires a C++ compiler, including though not limited to Stan.

- Download Xcode from the AppStore

However, since downloading this can be extremely time consuming given the large size of the full development tools suite an alternative option is to install a paired down version that provides the tools necessary for our purposes in this course without the overhead of the full Xcode development environment. You can install the paired down version of Xcode by running the code below in the RStudio console. Just copy and paste and hit enter.

```r
## MacOS Users may need to install the rstudioapi manually first
install.packages("rstudioapi")

## Run the command to install xcode-select
rstudioapi::terminalExecute(
  command = "xcode-select --install"
)
```

2.2 Quarto

If you have read through the course syllabus you will notice that I am having you work in something called Quarto. You may be asking your self what is Quarto I have only used MS Word. Quarto embraces something called literate programming. Essentially what this means is that words and code appear side by side. This guide was created in Quarto and what I use the most. Since we are going to produce lots of graphs and tables in this class the typical workflow for that would look something like this

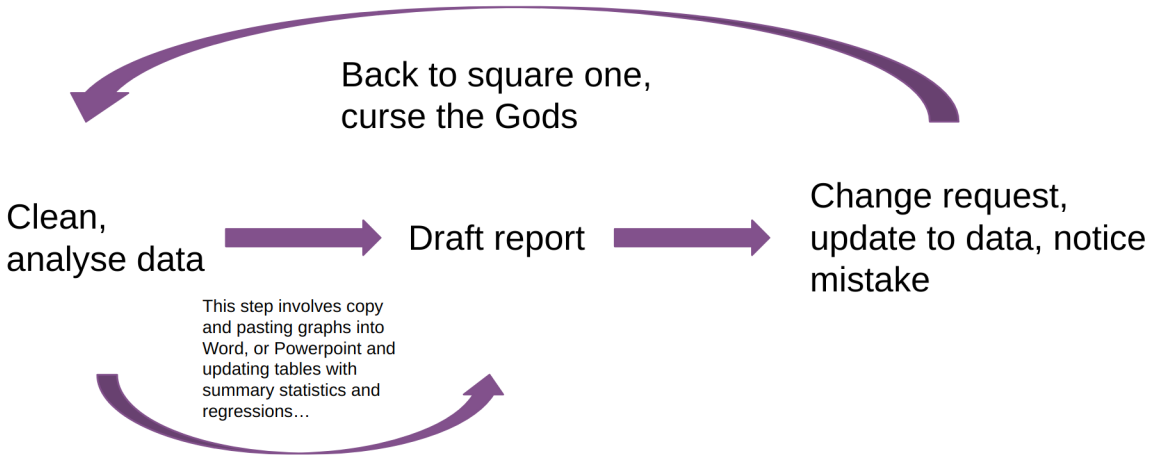


Figure 2.2: Credit for this image goes to Bruno Rodrigues

This involves lots of work for yourself. If you are doing a data analysis project where you produce 5 plots each time you make minor changes to those plots you are going to have to remember

where those plots are, copy and paste them over, resize them or reformat them. This gets infinitely more annoying if you are reporting numbers in tables or in text. In some cases data analysis teams are constantly updating reports for stake holders based on new data. So if you have a report that says our "based on our model we would expect an increase of 8 blah blahs" and later you rerun the analysis cuz there was new data or you notice a mistake you have to figure out where exactly you said "8 blah blahs" and switch them.

In Quarto this process is a whole lot easier. The loop looks like this

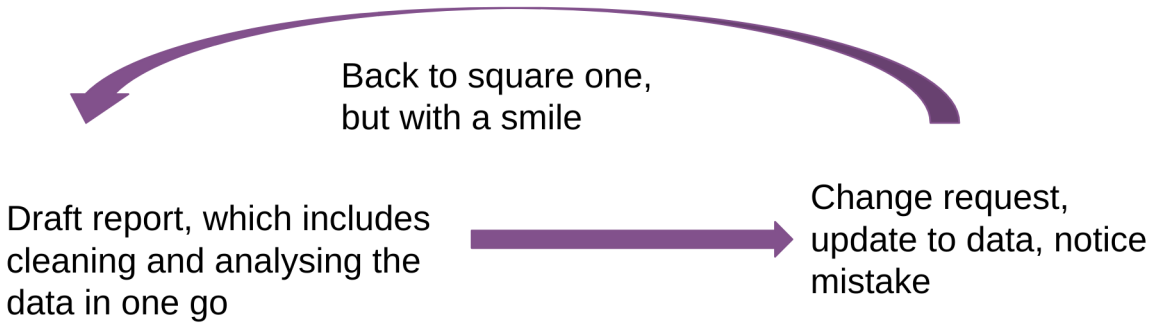


Figure 2.3: Credit for this image goes to Bruno Rodrigues

You are changing the code for your figures in the document itself. So any changes are going to appear in the document automatically! You can also use code inline to automatically update numbers!

```r
data <- 1:100

avg <- mean(data)
```

The average for our data is 50.5. In the document it looks like this

````
```{r}
data <- 1:100

avg <- mean(data)
```
````

```
```

The average of our data is `r avg`

You may have noticed that our data only goes from 1 to 100. We can make a quick modification and the document will update the document accordingly without any copy and pasting!

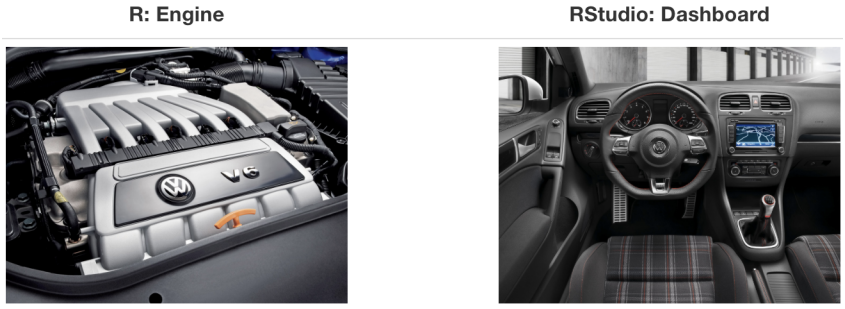
```
data <- 1:1000

avg <- mean(data)
```

The average of our data is 500.5

2.3 Navigating Rstudio

2.3.1 R? Rstudio? Whats the Difference?

- R is a statistical programming language
- RStudio is a convenient interface for R (an Integrated Developer Environment, IDE)
- At its simplest:
  - R is like a car's engine
  - RStudio is like a car's dashboard



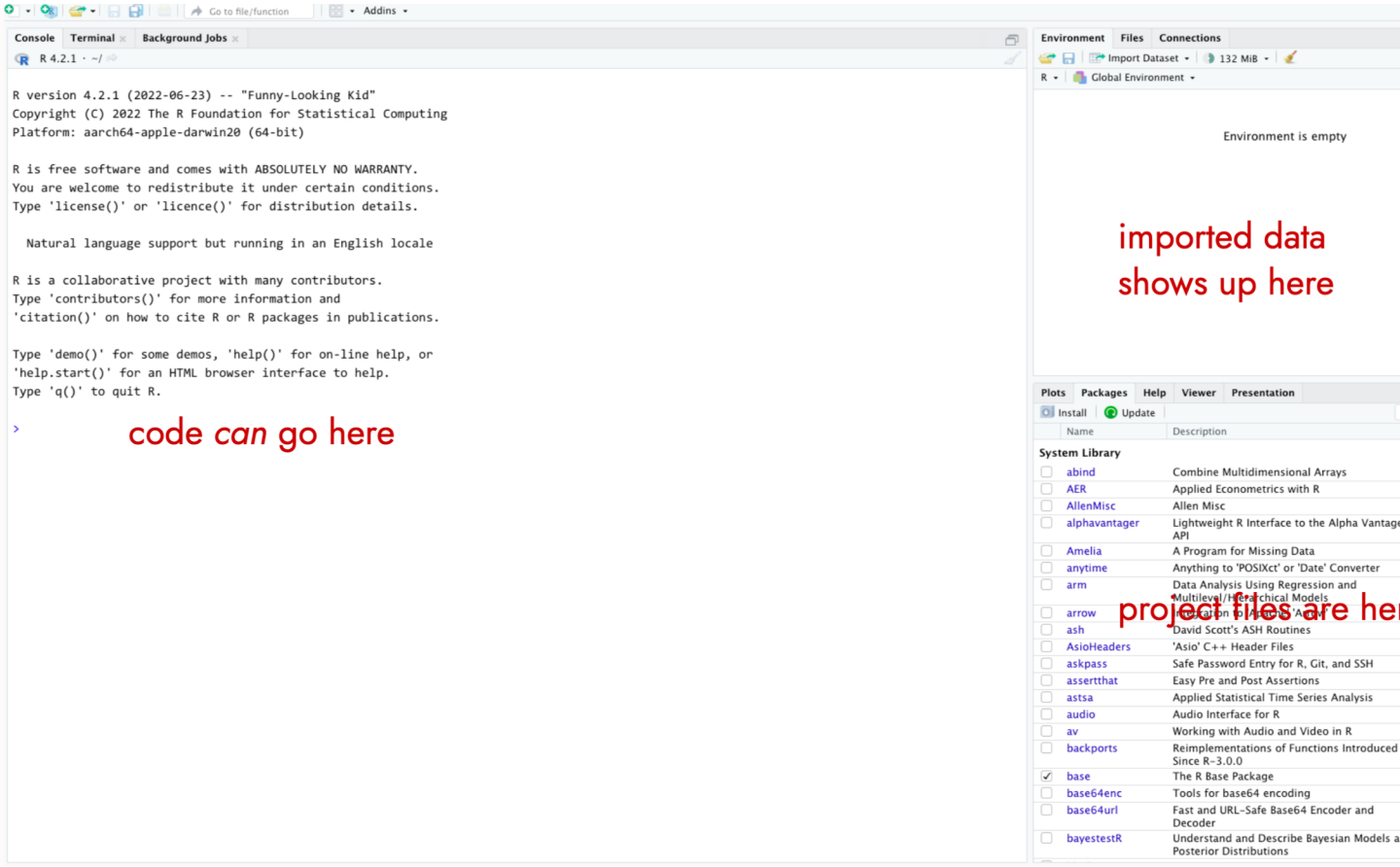
The most common way that we interact with R is through Rstudio you can technically run R by just opening R and typing in code. But most people do not do this. It is not especially

friendly to work in there is no syntax highlighting no code completetion. There isnt even really an option to add keyboard shortcuts. It is kind of like a Formula 1 car it can go real fast but it is not a comfortable drive.

Rstudio has lots of handy features that help you. Much like a car you and I would drive. If we didn't have the dashboard but still had the engine and some wheels and a steering we could drive the car if needed. However a car with a dashboard lets us figure out what the car is doing more easily.

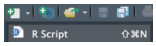
2.3.2 Navigating RStudio

When you open up Rstudio it should look something like this.

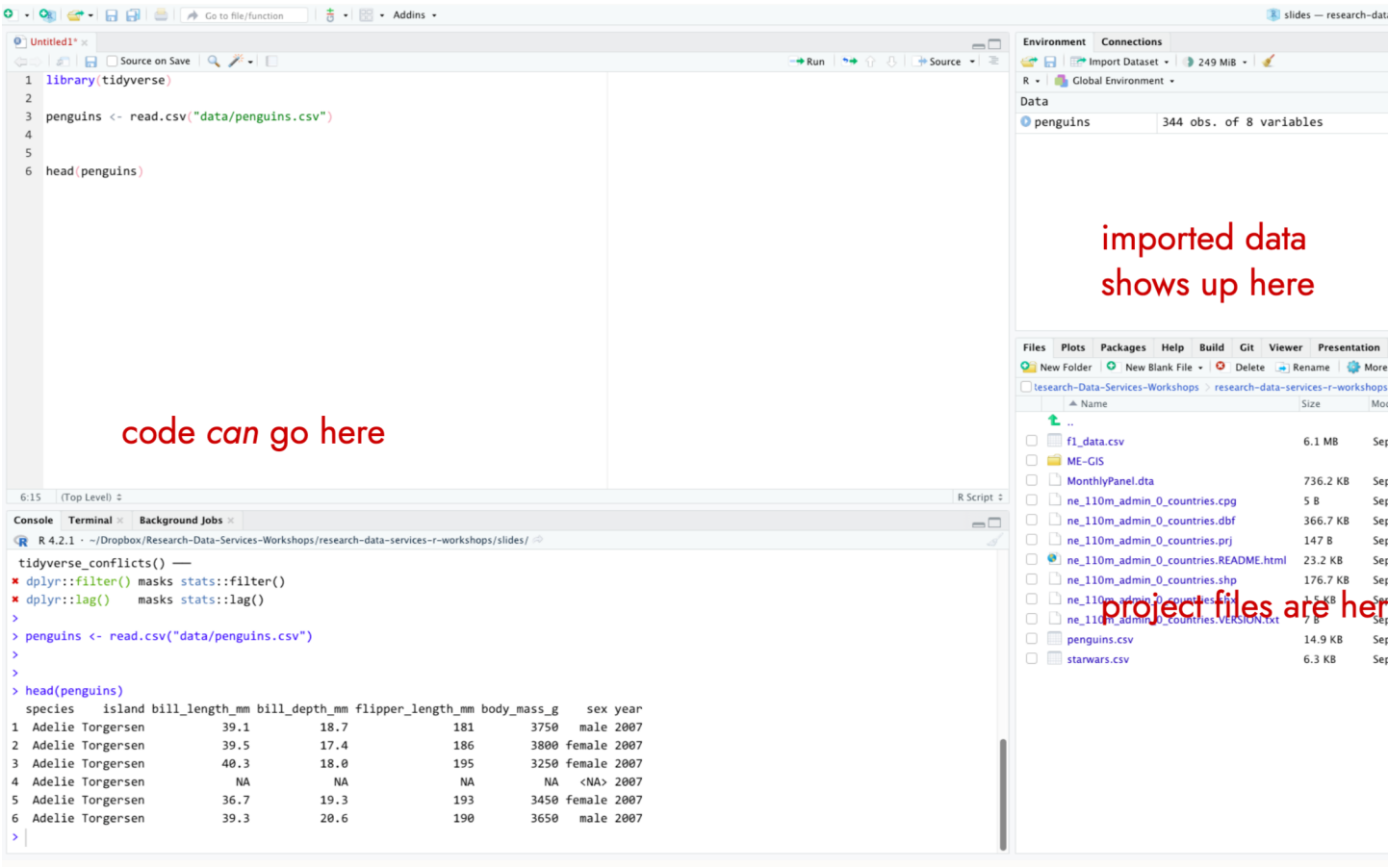


It is worth briefly walking through each pane. The code can go here part is called the `console`. If you just start typing R code and hit enter it will run. The `imported data shows up here` lets us peak at our `Global Environment`. This is where stuff we need/worked on shows up. `Project files are here` in the bottom right actually has a lot of panes there. The first one called `files` lets us peak at all the files in the folder that R is pointed at. The second pane `packages` lets us look at all the packages we have installed(more on this later). `Plots` lets us preview plots we have made.

Most of the time we don't enter code directly into the console. If we just enter code into the console we do not have a record of what we did! This is a bad workflow for lots of reasons! At a minimum you are making your life a whole lot more difficult because you will have to redo everything every single time. That is way more work for yourself. If you click on the little white page at the top left you will see something that looks like this.



This is a fairly standard way to store your code! This is a record of everything you did which is really important because it saves you time. Just as importantly it lets your colleagues see what you did to the data. If we don't know what you did than it is hard to know where things are going wrong or if what you did matches what you said.



If you opened an R script than it should look something like this. If you highlight the code and hit `cmd + enter` on a Mac or `ctrl + enter` if you are on a Windows then it will run the code for you!

Now that you have R Studio and R all set lets get started on the basics of R!

3 Basics of R

In this chapter I will introduce you to the basics of R and object oriented programming. These may be relatively new concepts for you and thats okay! I encourage you to follow along in Rstudio. Open up an R script and either type in the code

3.1 R as a fancy calculator

Good news you never have to buy a TI-84 again you can use R as a fancy calculator.

```
2+2 ## addition
```

```
[1] 4
```

```
4-2 ## subtraction
```

```
[1] 2
```

```
600*100 ##multiplication
```

```
[1] 60000
```

```
100/10 ##division
```

```
[1] 10
```

```
10*10/(3^4*2)-2 ## Pemdas
```

```
[1] -1.382716
```

```
log(100) # takes the log of 100
```

```
[1] 4.60517
```

```
sqrt(100) # sqrt of 100
```

```
[1] 10
```

However, if we want to reuse stuff we need to **assign them** to an object. This applies for all the numbers we calculated above and datsets you read in! This is somewhat unintuitive if you have never worked with an object oriented program. However, you will eventually get the hang of it!

3.2 Objects

3.2.1 Objects and Assignment

In the installation chapter I showed did some assigning without really telling you, but now it is time to explain what is going on. R is what is known as an object oriented programming language (OOP), meaning that everything in R is an "object." This is useful for understanding the concept of *assignment*, or how we specify certain objects in memory so we can use them elsewhere in a script. Values need to be assigned to objects using the assignment operator `<-`, `<` followed by `-`, and then on the right side you tell R what goes in that object. This can be as simple as specifying a calculation such as `4 + 4` or passing numeric values to a function such as `sum(9, 6)`.

> Tip
>
> The keyboard shortcut for `<-` is `alt + -`

```
numbs <- 4 + 4

chars <- "hello world"

numbs2 <- 3*3

numbs + numbs2
```

```
[1] 17
```

When we assign things the resulting values are not returned so we can "print" the object explicitly using `print` or referring to it by name.

```
print(numbs)
```

```
[1] 8
```

```
chars
```

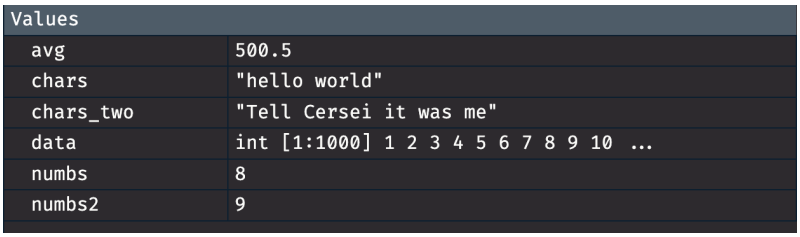
```
[1] "hello world"
```

One fairly common error you will run into in `R` is something along this line.

```
chars_two <- "Tell Cersei it was me"

charstwo
```

```
Error in eval(expr, envir, enclos): object 'charstwo' not found
```

While this is a fairly minor typo if we look at the global environment we will get a hint.



Values	
avg	500.5
chars	"hello world"
chars_two	"Tell Cersei it was me"
data	int [1:1000] 1 2 3 4 5 6 7 8 9 10 ...
numbs	8
numbs2	9

When we asked `R` to print `charstwo` it is going to look in the global environment for something called `charstwo` to print and if it can't find it then R will give up. We don't have anything called `charstwo` but we do have something called `chars_two`. We know what we meant when we made the typo, but the computer does not! Computers are really good at doing things, but they require really explicit instructions so if we type `chars_two` it will work.

```
chars_two
```

```
[1] "Tell Cersei it was me"
```

One way to make life a little bit is have a consistent naming convention! This will generally help you because you will develop some tendencies. However, this is an error we all still get because it is easy to forget what all your objects are named. You can either look at your global environment or scroll up and see what past you named the object.

3.2.2 Naming Objects in R

The best practice is to use concise descriptive names.

When loading in data typically I do `raw_my_dataset_name` and after data all of my cleaning I do `clean_my_dataset_name`

- Objects must start with a letter. But can contain letters, numbers, `_`, or `.` There are a few different naming conventions that people use.

- snake_case_like_this_is_what_I_use
- somePeopleUseCamelCase
- some_People.are_Do_not.like_Convention[1]

3.2.3 Things we can never name stuff in R

3.3 if/else

`if` used when something meets a certain condition the code will do something. `else` is used when the if condition is not met

3.4 for/while

`for` is how we construct something called a for loop. `while` is used to construct something called a while loop. These are ways to repeat the same task over and over again and are building blocks of packages in R.

3.5 function

How we specify user defined functions. This is building block of programming in R

3.6 TRUE/FALSE

Logical constants in R

3.7 NULL

is returned when an expression or function results in an undefined value

[1] Example and Discussion provided in R for Data Sciency by Hadley Wickham