

Getting Started with Data (in R)

Jonathan Moreno-Medina

ECO3253, UTSA

Fall 2022

Why R again?

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**.

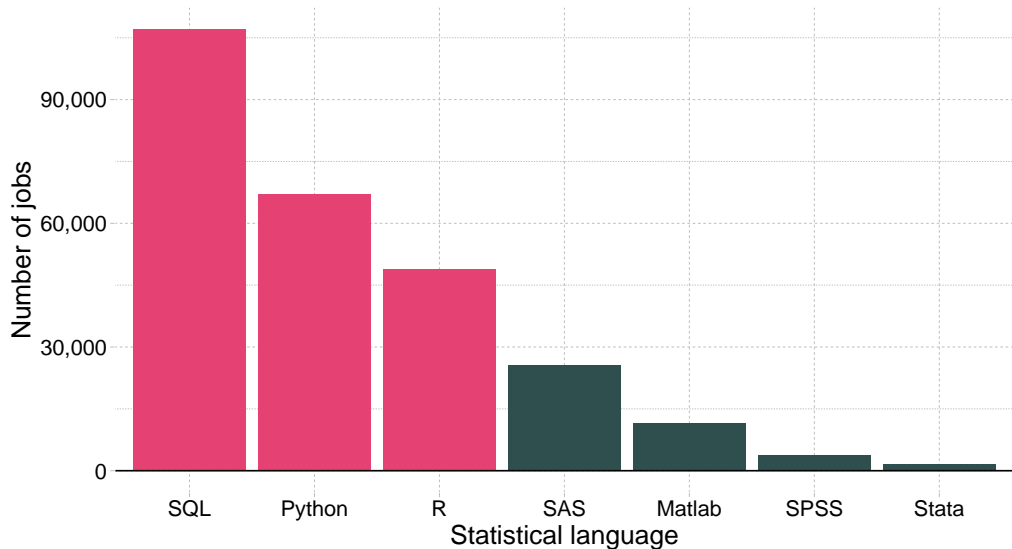
Ok, but why R?

- ▶ **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

Added benefits of learning R

Comparing statistical languages

Number of job postings on Indeed.com, 2019/01/06



Getting Started with Data in R

Basic questions

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 will cover those 3 points today, and you will start exploring your first dataset based on what we covered!

What are R and RStudio?

We will use R via RStudio. They are not the same!

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

R: Engine



RStudio: Dashboard



What are R and RStudio?

- ▶ R: programming language that runs computations
- ▶ RStudio: *integrated development environment (IDE)* - interface that add 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.

R and RStudio: In your computer or in the cloud?

To use R and RStudio, you can:

- ▶ install it in your computer (see the book) or ...
- ▶ run it on someone else's computer (the cloud!). We will do that, so you don't have to worry about installations.

Let's open R!

RStudio in the cloud

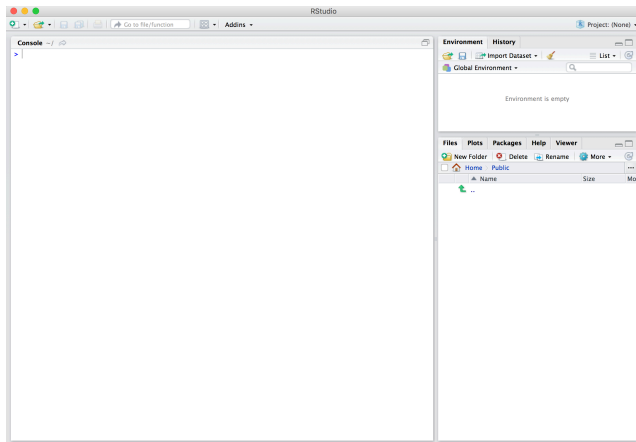
Let's jump in! You should have received a link for you to access RStudio in the cloud. The Department of Economics is paying for us to use this service for this class. Remember that all the **code is running on someone's computer**. In this case, it is running on a computer owned by the folks at RStudio.

INCLUDE FIGURE OF RStudio Cloud

Starting a project

Include Figure

Using R via RStudio



When open RStudio, you should see this (<-)

3 panes (panels dividing the screen):

1. The *Console pane*
2. *Files pane*
3. *Environment pane*

We will soon learn what they are for.

Simple coding in R

How do I code in R?

“OK. Now how do I use R?” 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.

That sounds scary if you have not programmed, but turns out **programming in R is fairly simple**

Basic programming concepts and terminology - 1

You'll “learn by doing.” Whenever you see the following, it means `computer_code`, not normal text.

How will you learn? **Do the HOMEWORKS with time...** I cannot emphasize this enough!

- ▶ 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.

Basic programming concepts and terminology - 2

- ▶ *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 will use this a lot!

Basic programming concepts and terminology - 3

► *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 operators 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*.

In Class Exercise

```
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 'vec
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
```

Errors, warnings, and messages

What is scare for most new R and RStudio users? *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

- ▶ **Errors:** When **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 @ref(package-use) 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

- ▶ **Warnings:** When **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, we will see later that 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

- ▶ **Messages:** When **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 section or when you read data saved in spreadsheet files with the `read_csv()` function. 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()`.

What to do if you see red text?

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.

Tips on learning to code

Learning to code/program is like learning a foreign language: can be daunting and frustrating at first. But as learning a foreign language, effort + not afraid to make mistakes = anybody can learn.

Tips:

- ▶ **Remember that computers are not actually that smart:** You have to tell a computer everything it needs to do. Instructions should not have mistakes nor be ambiguous.
- ▶ **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.
- ▶ **The best way to learn to code is by doing:** Do the homeworks with time! This are opportunities for you to try and get familiar with coding.
- ▶ **Practice is key:** Just as the only method to improving your foreign language skills is through practice, practice, and practice.

R Packages

What are R packages?

Usual confusion with many new R users: packages. They give extra functionality to R (extra 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 in the course are the `ggplot2` package for data visualization. We will see more on how to use `ggplot2` and `dplyr` package (helpful to edit and modify data - or wrangling).

R packages = apps for a phone:

R: A new phone



R Packages: Apps you can download



Packages as Apps

How do you open an app like Instagram on your phone?

1. *Install the app*: New phone does not have Instagram app -> you need to download it. 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.

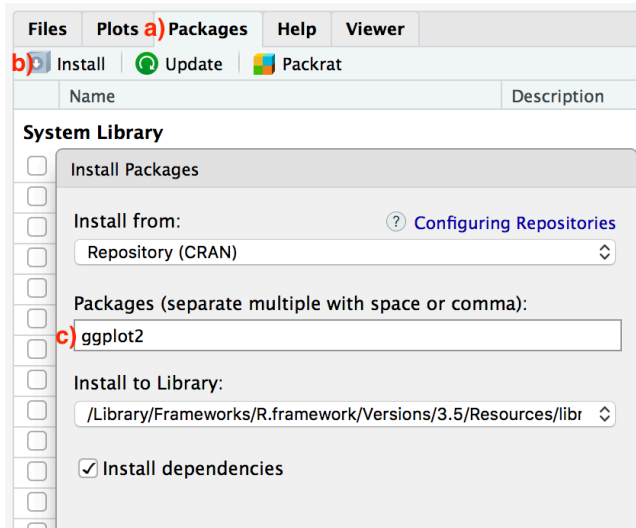
Once Instagram is open on your phone, you can then proceed to share your photo. Almost the same for an R package. You need to:

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. If you want to use a package for the first time, you need to install it first. Once you've installed a package, unlikely you need to install again.
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 install ggplot2 package for data visualization.

Package installation

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



► **Easy way:** In the Files pane of RStudio:

- a) Click on the “Packages” tab
- b) Click on “Install”
- c) Type the name of the package under “Packages (separate multiple with space or comma):” In this case, type `ggplot2`
- d) Click “Install”

Package installation - 2

► **Slightly harder way:** A more common way to install a package is by typing

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

in the Console pane of RStudio and hitting enter. Note you must include the quotation marks.

Excercise - Install Packages

Install packages `dplyr`, and `knitr` packages. This will install the earlier mentioned `dplyr` package, and the `knitr` package for writing reports in R.

Package loading

Recall: after installing a package, you need to “load” it (open it). We do this by using the `library()` command. For example, to load the `ggplot2` package, run the following code in the Console pane. What do we mean by “run the following code”? Either type or copy & paste the following code into the Console pane and then hit the enter key.

```
library(ggplot2)
```

After running above code, do you see blinking cursor returns next to the `>` “prompt” sign? YES: success! `ggplot2` package is now loaded and ready to use; NO: got a red “error message” that reads...

```
Error in library(ggplot2) : there is no package called ‘ggplot2’
```

... it means that you didn’t successfully install it. In that case, go back to the previous subsection “Package installation” and install it.

Excercise - Load Packages

Load packages `dplyr`, and `knitr` packages as well by repeating the above steps.

Package use

One extremely common mistake new R users make when wanting to use particular packages is that they forget to “load” them first by using the `library()` command we just saw. Remember: *you have to load each package you want to use every time you start RStudio*. If you don’t first “load” a package, but attempt to use one of its features, you’ll see an error message similar to:

```
Error: could not find function
```

R is telling you that you are trying to use a function in a package that has not yet been “loaded.” Almost all new users forget to do this when starting out, and it is a little annoying to get used to. However, you’ll remember with practice.

Hands-on exercise!

Explore your first dataset: economic mobility in the US

Let's put everything we've learned so far into practice and start exploring some real data! These “spreadsheet”-type datasets are called *data frames* in R; we will focus on working with data saved as data frames throughout this course.

Step 1: Load all the packages needed for this exercise (assuming you've already installed them).

Economic mobility data

The [Opportunity Atlas](#) is a freely available interactive mapping tool that traces the roots of outcomes such as poverty and incarceration back to the neighborhoods in which children grew up. The atlas dataset we loaded has the underlying data to describe equality of opportunity across the 73,278 different neighborhoods in the United States.

Let's unpack these data a bit more!

atlas data frame

We will begin by exploring the atlas data frame we just loaded to get an idea of its structure. Run the following code in your console (either by typing it or cutting & pasting it): it loads the atlas dataset into your Console. Note depending on the size of your monitor, the output may vary slightly.

```
atlas
```

```
# A tibble: 73,278 x 62
```

	tract	county	state	cz	czname	hhinc_mean2000	mean_commutetime2000
	<dbl>	<dbl>	<dbl>	<dbl>	<chr>	<dbl>	<dbl>
1	20100	1	1	11101	Montgomery	68639.	26.2
2	20200	1	1	11101	Montgomery	57243.	24.8
3	20300	1	1	11101	Montgomery	75648.	25.3
4	20400	1	1	11101	Montgomery	74852.	23.0
5	20500	1	1	11101	Montgomery	96175.	26.2
6	20600	1	1	11101	Montgomery	68096.	21.6
7	20700	1	1	11101	Montgomery	65182.	23.2
8	20801	1	1	11101	Montgomery	76874.	30.3

Exploring data frames

Among the many ways of getting a feel for the data contained in a data frame such as `atlas`, we present three functions that take as their “argument”, in other words their input, the data frame in question. We also include a fourth method for exploring one particular column of a data frame:

1. Using the `View()` function built for use in RStudio. We will use this the most.
2. Using the `glimpse()` function, which is included in the `dplyr` package.
3. Using the `$` operator to view a single variable in a data frame.

View() - Part 1

1. View():

Run `View(atlas)` in your Console in RStudio, either by typing it or cutting & pasting it into the Console pane, and explore this data frame in the resulting pop-up viewer. You should get into the habit of always *Viewing* any data frames that come your way. Note the capital “V” in `View`. R is case-sensitive so you’ll receive an error if you run `view(atlas)` instead of `View(atlas)`.

(LC2.1) What does any *ONE* row in this `atlas` dataset refer to?

- ▶ A. Data on an neighborhood
- ▶ B. Data on a state
- ▶ C. Data on an person
- ▶ D. Data on multiple neighborhood

View() - Part 2

By running `View(atlas)`, we see the different *variables* listed in the columns and we see that there are different types of variables. Some of the variables like `poor_share2010`, `hhinc_mean2000`, and `share_hisp2010` are what we will call *quantitative* variables. These variables are numerical in nature. Other variables, like `tract` are *categorical*: they are just names (even if they have numbers). For example `tract` represents a Tract FIPS Code, that is, a 6-digit code assigned by the census folks to each neighborhood in 2010.

Note that if you look in the leftmost column of the `View(atlas)` output, you will see a column of numbers. These are the row numbers of the dataset. If you glance across a row with the same number, say row 5, you can get an idea of what each row corresponds to.

glimpse()

2. glimpse():

The second way to explore a data frame is using the `glimpse()` function included in the `dplyr` package. Thus, you can only use the `glimpse()` function after you've loaded the `dplyr` package. This function provides us with an alternative method for exploring a data frame:

```
glimpse(atlas)
```

Rows: 73,278

Columns: 62

```
$ tract <dbl> 20100, 20200, 20300, 20400, 20500, 20600
```

```
$ county <dbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 3
```

```
$ state <dbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1
```

```
$ cz <dbl> 11101, 11101, 11101, 11101, 11101, 11101
```

```
$ czname      <chr> "Montgomery", "Montgomery", "Montgome
```

```
$ hhinc_mean2000      <dbl> 68639, 57243, 75648, 74852, 96175, 680
```

```
$ mean_commutetime2000      <dbl> 26.2, 24.8, 25.3, 23.0, 26.2, 21.6, 23.8
```

glimpse() - Part 2

We see that `glimpse()` will give you the first few entries of each variable in a row after the variable. In addition, the *data type* of the variable is given immediately after each variable's name inside `< >`. Here, `int` and `dbl` refer to “integer” and “double”, which are computer coding terminology for quantitative/numerical variables. In contrast, `chr` refers to “character”, which is computer terminology for text data. Text data, such as the `czname` (the name of the metro area), are categorical variables.

\$ operator

3. \$ operator

Lastly, the \$ operator allows us to explore a single variable within a data frame. For example, run the following in your console

```
atlas$tract
```

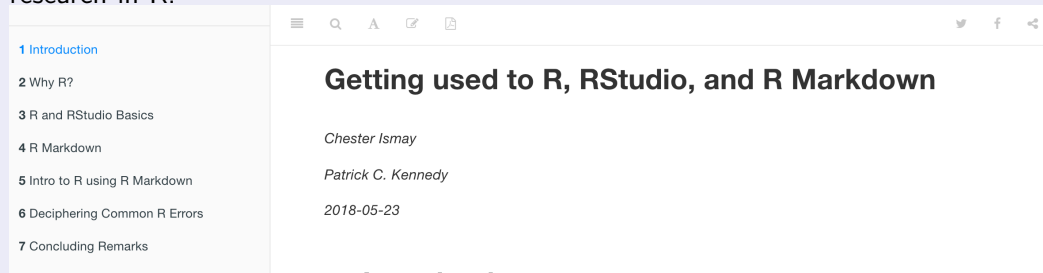
We used the \$ operator to extract only the tract variable and return it as a vector of length 73,278. We will only be occasionally exploring data frames using this operator, instead favoring the `View()` and `glimpse()` functions.

Conclusion

We've given you what we feel are the most essential concepts to know before you can start exploring data in R. There is much more to explore in R but this is a great place to get started!

Additional resources

If you want to dive more and feel you could benefit from a more detailed introduction, check this short book: [Getting used to R, RStudio, and R Markdown](#) short book. It has screencast recordings that you can follow along and pause as you learn. Furthermore, there is an introduction to R Markdown, a tool used for reproducible research in R.



The screenshot shows a digital book interface. On the left is a table of contents with seven items: '1 Introduction' (highlighted in blue), '2 Why R?', '3 R and RStudio Basics', '4 R Markdown', '5 Intro to R using R Markdown', '6 Deciphering Common R Errors', and '7 Concluding Remarks'. The main content area on the right displays the title 'Getting used to R, RStudio, and R Markdown' in a large, bold, black font. Below the title, the authors 'Chester Ismay' and 'Patrick C. Kennedy' are listed, followed by the date '2018-05-23'. At the top of the main area is a navigation bar with icons for a menu, search, font size, editing, and a document icon. On the far right of this bar are social media icons for Twitter, Facebook, and a share icon. In the bottom right corner of the entire image, there are three small, light blue navigation icons: a back arrow, a magnifying glass, and a circular refresh icon.

1 Introduction	Getting used to R, RStudio, and R Markdown
2 Why R?	<i>Chester Ismay</i>
3 R and RStudio Basics	<i>Patrick C. Kennedy</i>
4 R Markdown	2018-05-23
5 Intro to R using R Markdown	
6 Deciphering Common R Errors	
7 Concluding Remarks	