PS 312: Programming with R
Course Notes

Abhijit Dasgupta, PhD

Last updated: March 24, 2019

Welcome

This course is an introduction to the statistical programming language R and various applications. We will cover the entire data analytics pipeline from data ingestion to data wrangling, summarizing, modeling, visualizing and reporting, all using tools found within the R ecosystem.

The version of these notes you are reading now was built on 2019-03-24.

Reproducibility

These notes are written with `bookdown`, a R package for writing books using `rmarkdown`. All code in these notes were developed on R version 3.5.0 (2018-04-23), using the same packages pre-installed in your virtual machines. When you're on your own, you will need to install a recent version of R, and also install the corresponding packages, on your computer, for all the code to work. A listing of all the packages used in this course will be available as an appendix.

To build these notes locally, clone or download the Github repo hosting these notes, unzip it if necessary, and double-click on `FSI_Book.Rproj`. Assuming you have RStudio installed, this will open this project (more on *RStudio Projects* later). You can then go to the console and enter the following code:

```
bookdown::render_book("index.Rmd") # to build these notes
browseURL("_book/index.html") # to view it
```

Starting up

Chapter 1

What is R?

R is the most popular[1] open source *statistical programming language* in the world. It allows you to

1. read datasets written in a wide variety of formats,
2. clean and process the data,
3. derive summaries,
4. run analytics,
5. visualize
6. create automated reports, presentations, websites, dashboards and interactive applications

R is not just a language, but an ecosystem comprising over 15,000 user- and corporation-developed *packages* or modules, all written in the R language for a variety of purposes. It is a very flexible and customizable language, which is why it is used by an estimated 2 million users worldwide for data analytics. The question R users often ask is not "Can it be done?" but rather "How can it be done?". R is used in areas as varied as healthcare, economics, forestry, oceanography, pharmaceuticals, artificial intelligence and natural language processing.

Why is R so widely used? Some reasons are:

- R is open source, so it is accessible to anyone with a computer
- Since the code in R and all its packages are open, the community of users can help debug it and make it more reliable and robust
- The R ecosystem is very rich in tools for doing data analytics in particular, so there is almost certainly something available for almost any task
- The community of R users worldwide is a very strong, well-connected group who are welcoming, ready to help, cooperative and inclusive. Many users find this community to be one of the most attractive things about R
- R produces really nice customizable visualizations with relatively little effort, which was one of the first reasons for popularity.

[1] https://spectrum.ieee.org/static/interactive-the-top-programming-languages-2018

A note on coding and programming

R does not have a *point-and-click* interface that you are probably more familiar with from Excel, Word or other computer applications. It requires you to *code*, i.e. write instructions for the computer to, in the case of R, read, analyze, graph and report on datasets.

R is first and foremost a **language**. So, instead of thinking that this is some geeky thing that "programmers" and "IT people" do, think of it as learning a language. You will see that, like any language, it has nouns, verbs, adjectives and adverbs, and you can create "sentences" that start with data and end in something useful like a table, graph or document. With a traditional spoken and written language like French, Arabic, Farsi or Japanese, you learn it to be able to interact with people at different posts around the world. With a programming language like R, you will be able to interact with **data**, to make sense of it, to describe it, and to present it.

Coding

Coding is writing explicit instructions to a very literal, and in some ways, stupid machine. The machine takes our code literally, and will do **exactly** what you tell it to do in the code. If you are getting unexpected results, it's almost certainly your code that needs to be checked, not the machine.

R, the language

As we will see, R has many elements of a language.

- **Objects**: These are the *nouns*. We will act on objects to create new objects. Each object has a *name* which we will treat as the nouns in our code.
- **Functions**: These are the *verbs*. Functions will act on objects to create new objects.
- **The %>% operator**: This acts like the conjunction "then" to create "sentences" called *pipes* or *chains*.
- **Optional function arguments**: These are adverbs which modify the action of the function (verb).

While writing code in R, we should be aware that R is **case-sensitive**, so `mydata` is a different object than `myData` which is also different from `Mydata` and `My_Data` and `MyData` and `my_data`.

The ultimate goal for every script file is to create a "story" using the language of R, starting from data to create descriptions, understand patterns through visualization and modeling, and analyzing the data in general. Scripts make this story reproducible, and also transferable to different data sets.

Of course, as with any beginner writer, your coding will be sloppy at first, will suffer many stops and starts and strike-throughs and modifications and throwing things into the proverbial trash. With practice, this will become easier and smoother and more effective and more expressive. This workshop is designed to give you an initial push towards that goal.

So, let's start this journey.

R Packages

Packages are modules of R code that enhance the capabilities of R. Many packages are well established and curated, and have to go through a strict software compatibility review before allowed on CRAN.

Installing packages

Installing packages on your computer can be done from the RStudio menu (`Tools > Install Packages`), or by running the command `install.packages(<package name>, repos = "https://cran.rstudio.com")`. For example, to install the `readxl` package, which we will use shortly, we would run the code

```
install.packages("readxl", repos="https://cran.rstudio.com")
```

> You can set the default repository in RStudio, in `Tools > Global Options`.
>
> Be aware that everything here is case-sensitive

Another way to install packages is to go to the `Packages` pane in RStudio, use the search bar there to find the package you want to install, and then click the checkbox beside the name. This is convenient, but not very reproducible if you have to move to a different computer, so it's generally discouraged.

How do you find packages? Glad you asked. The easiest way to find packages on CRAN is actually through the RStudio `Packages` pane, where the entire set of available packages are listed with a brief, top-line description. You can click on the package name to see a much more detailed overview of the packages, and many packages do have vignettes which give more information. However, once you've found the package you want, you should really code it up with `install.packages`, so that you can save the script for later when you might need to remember it again.

Loading packages in R

We will use several packages to enhance our experience and get going faster. However, to use a package, you must first load it into R. To load a package into R, you use the function `library` (ironically).

The first package we will load is the `tidyverse` package. This is actually a meta-package, which in turn loads a bunch of other packages. These form a core group of useful packages that are widely used, including

- readr (reading data from text files)
- tidyr (Manipulation, pivoting)
- dplyr (summarize, aggregate, filter)
- ggplot2 (visualization)
- purrr (functions applied across data structures, meta-programming)
- stringr (string manipulation)
- forcats (categorical data)

In addition, we'll load the `readxl` package for reading Excel files.

```
library(tidyverse)
library(readxl)
```

R Resources

There are many high quality resources for learning R available online. This is a selection of what I find most useful.

1. CRAN Task Views: These are curated lists of R packages for various purposes, ranging from econometrics to mathematics, finance, imaging, social sciences, time series, spatial analyses and more.
2. RStudio Cheatsheets: These are high-quality cheatsheets about different aspects of the R analytic pipeline.
3. StackOverflow #r: The `r` tag on StackOverflow is *the* place to find answers about R
4. Twitter #rstats: The who's who of R hang out at the #rstats hashtag, and questions can get answered very quickly. Also a way to find out what new packages are coming up
5. R-Bloggers: A blog aggregator which collects a few hundred R-related blogs in one place (including mine, in the interests of disclosure)
6. RSeek: When one realizes that R is just a letter in the alphabet, Google searches can be a bit difficult. RSeek has created a custom search targeted at R-related topics, sites and packages on the web.

Chapter 2

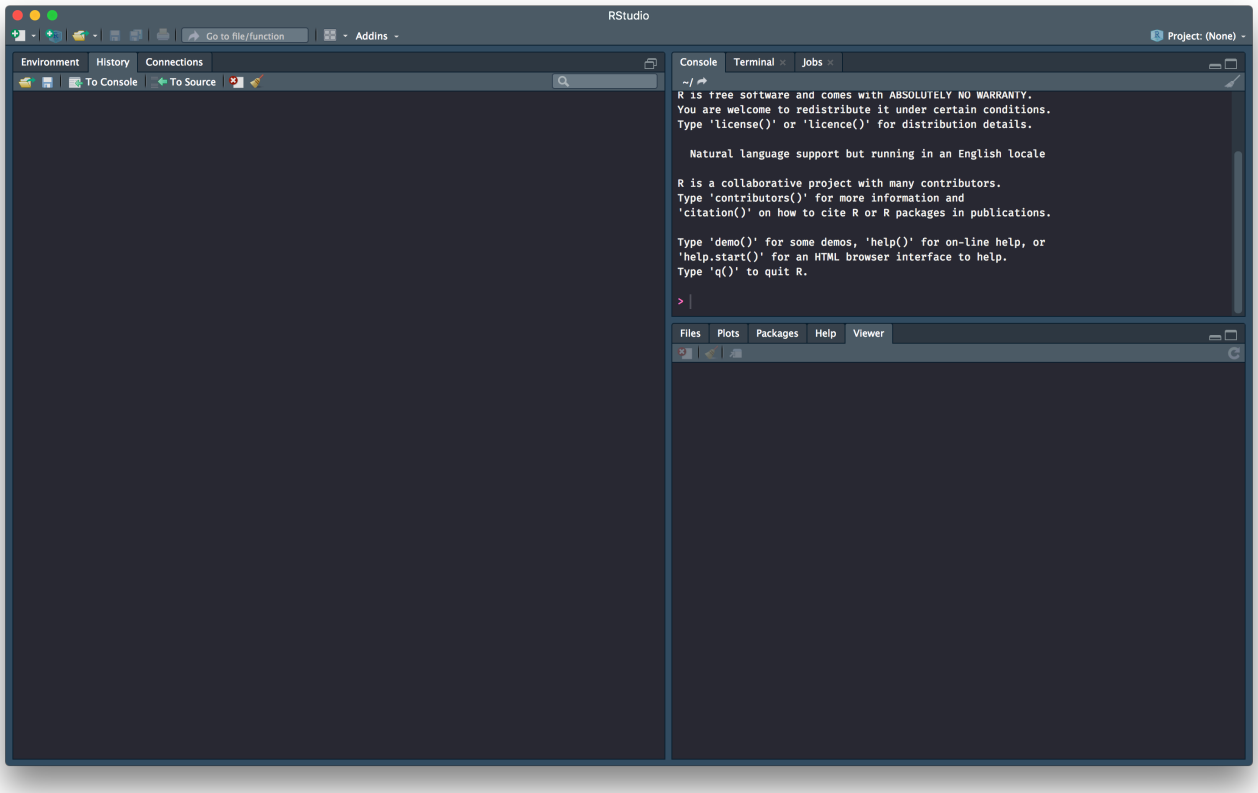
RStudio, your development environment (authoring program)

While R is the language we will learn, RStudio is the interface (or *integrated development environment*) we will use to write it, interact with it, and see our results. RStudio provides by far the most user-friendly interface to R (though it's not point-and-click).

This is the "journal" or "notebook" in which you will start your writing journey in R

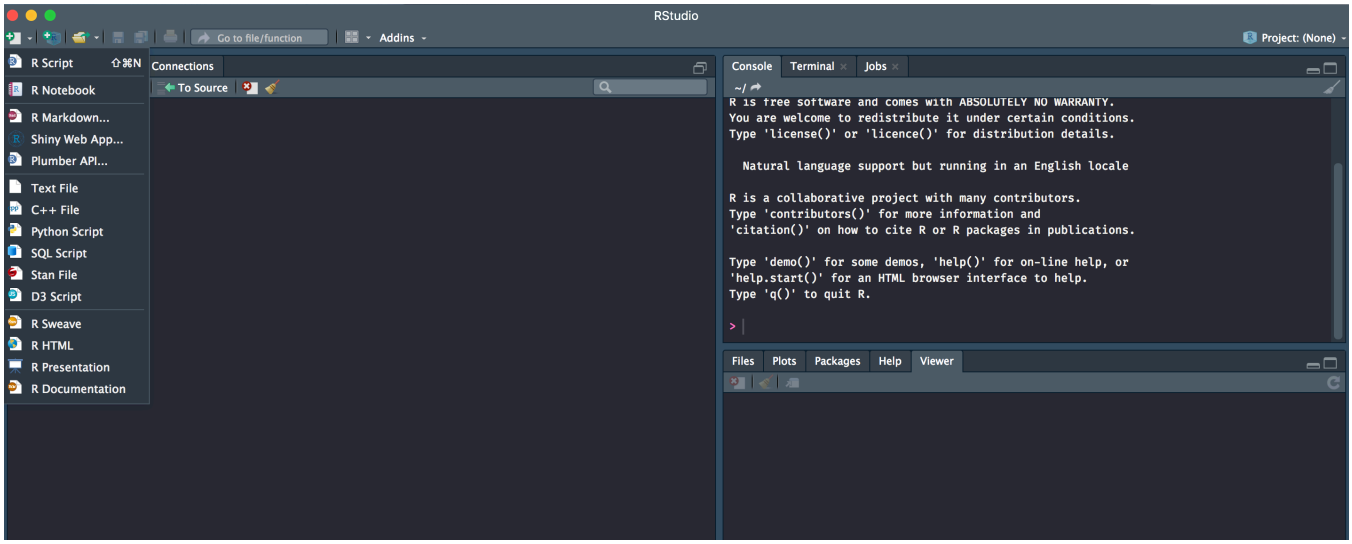
Starting RStudio

When you open RStudio, either from your desktop or from the Start menu, you'll see something like this:

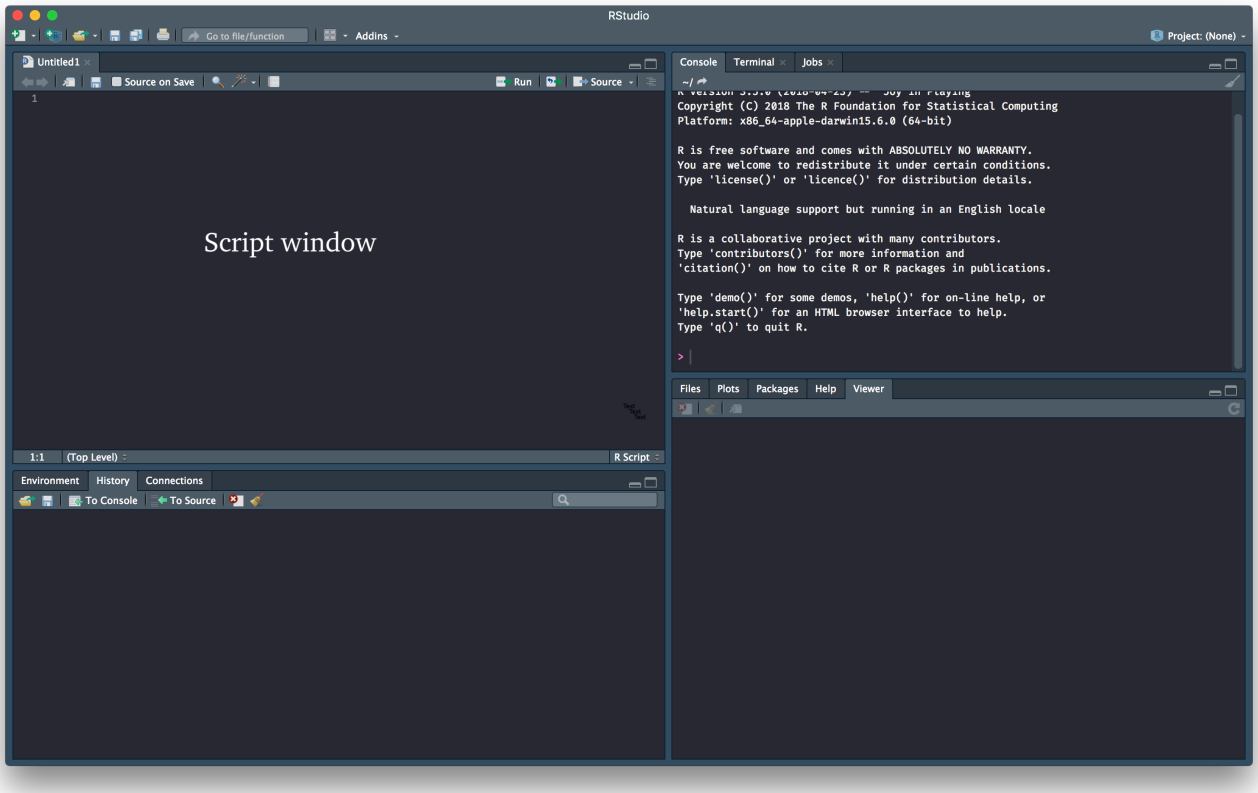


> I'll note that I've done some customization to my console, which you can also do by going to `Tools > Global Options`. Your screen will most likely have a white rather than a dark background

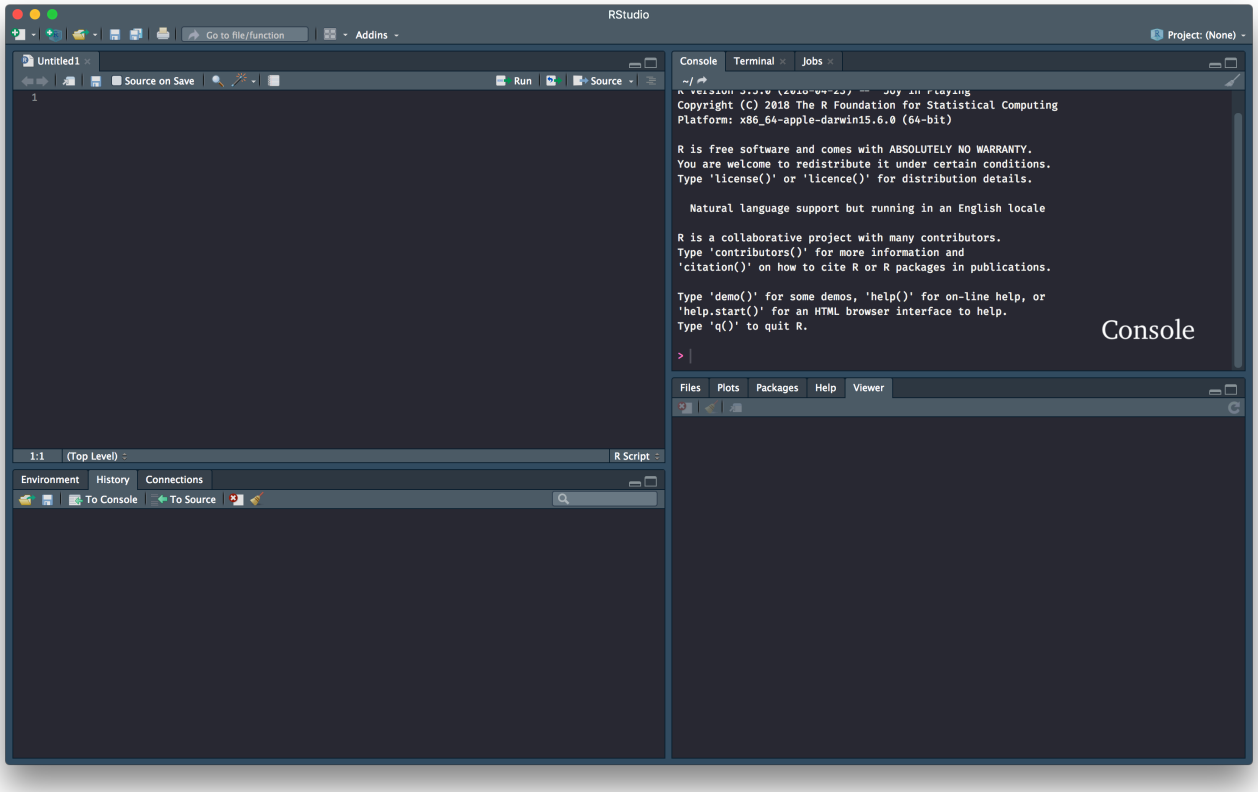
You can open a new panel for an R script using either `File > New File > R Script` or using the button at the top left of the window:



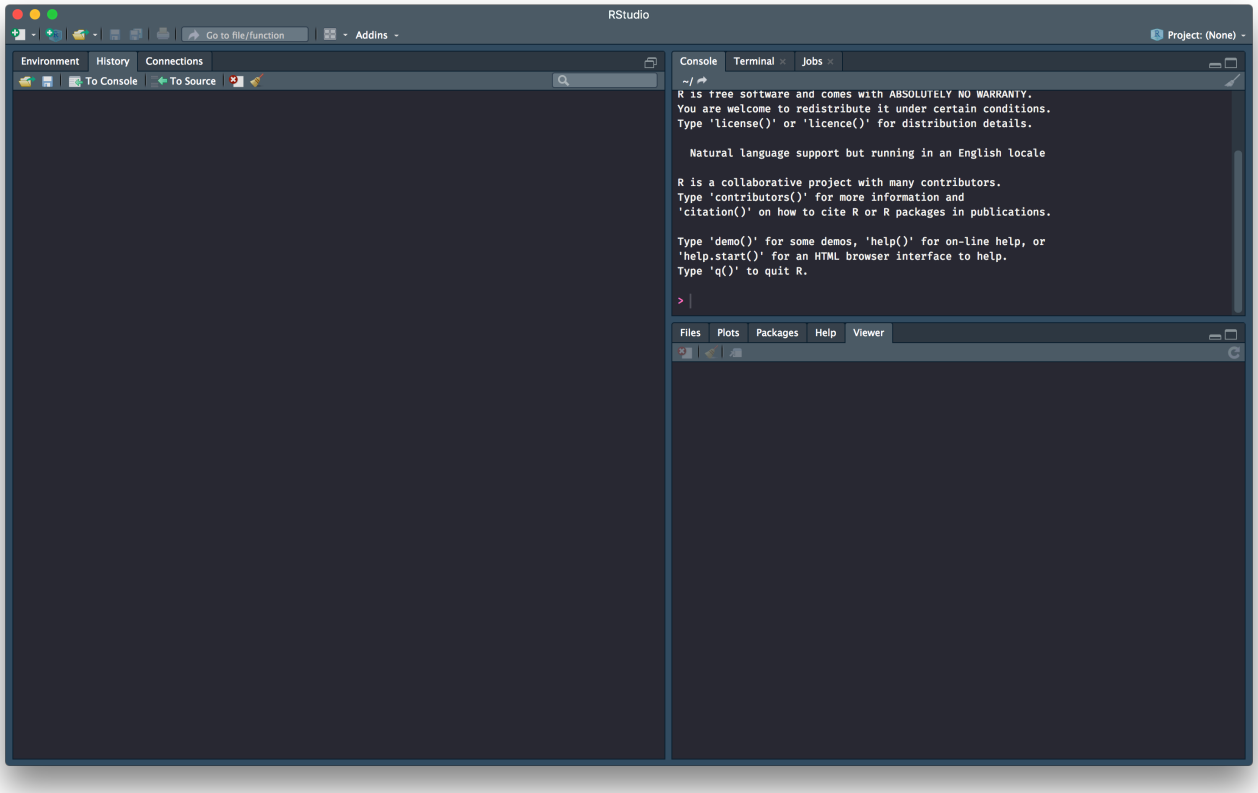
This opens up an R script file that you can edit and save. You will mainly be writing in this panel within a R script (see 2.1 for more details).



You will also have a Console panel where the code will actually run in R.



Other panes



There are several other panes in RStudio that we will see in due course.

- **Environment**: This shows all the objects ("words") in your current environment
- **History**: This gives a history of the commands you have run. This is searchable. Though you do have a stored history, see 2.1 for why you shouldn't fall to temptation to just code in the console.
- **Files**: This is exactly like File Explorer in Windows, and lets you see the contents of a folder/directory
- **Plot**: These is where the plots will show up. See **??** for more details on how to create plots
- **Packages**: This gives a listing of installed packages. You *can* click on the tick boxes to load packages into your environment, but I prefer coding it in (see section **??**) to make it reproducible and verifiable.
- **Help**: This will show help files once they are evoked
- **Viewer**: This pane shows results when they are produced as HTML documents. This pane will also come into play once we start with interactive visualizations in section **??**.

Feel free to explore these different panes and understand their functionalities.

2.1 Rstudio workflow

As we've seen, RStudio has both a scripting pane to write code, and a console pane to run code. Of course, you can write code directly into the console, but it is **not** a good practice. You will tend to get sloppy, lose the "story", and generally have less reproducible code.

Writing the program ("story") is just more reliable if you write into the script file and the send it to the console

to run. Sending it to the console can be acheieved with a keyboard shortcut, Ctrl-Enter (or Cmd-Enter on a Mac). This is something that will be second nature while coding in RStudio.

When you write code, be sure to comment your code liberally. In R, any line or any phrase starting with # is considered a comment and is ignored by the program. This allows you to comment your code, explain your ideas to yourself and generally make your code more readable. To further this goal, write your code in differnt lines, and indent, to make it more readable; R ignores white space in your file.

Why bother doing this? Basically because the most likely next person to see your code is going to be you in 6 months, and you don't want to be scratching your head wondering what you were doing earlier (been there, done that, don't like it). You certainly can't phone your earlier self, so the best strategy is to write comments for your future self to minimize future grief.

Using R to access data

Chapter 3

Loading data into R

R can access data files from a wide variety of sources. These include

1. Text files (csv, tsv, fixed-width)
2. Microsoft Excel files
3. Microsoft Access databases
4. SQL-based databases (MySql, Postgresql, SQLite, Amazon Redshift)
5. Enterprise databases (SAP, Oracle)

The R package `rio` can help read and write to many file types that are single files, and the package `rodbc` can do the same for the databases.

Exercise: Install the R package `rio` into your R installation

```
install.packages("rio", repos = "https://cran.rstudio.com") # Note the quotes
```

```
##
## The downloaded binary packages are in
##  /var/folders/5j/5lprmgt930xdpdn014f8st280000gn/T//RtmphfQd6T/downloaded_packages
```