A Progressive Introduction to Linear Models

Joshua French

Contents

Preliminaries

I designed this book to progressively introduce you to the analysis of data using linear models. My goal is to provide you with the skills needed to perform a linear regression analysis sooner rather than later. Some material that could be covered together (for example, all the different types of statistical tests and confidence intervals) has been broken into two sections: an early one to give you foundational knowledge about the topic and then later material to advance your understanding. Most of the detailed derivations have been placed in **Going Deeper** sections or in their own chapter, which can be skipped over to more quickly progress through the material if you do not want to focus as much on theory.

Acknowledgments

The **bookdown** package (Xie 2023) was used to generate this book. The **kableExtra** package (Zhu 2021) was to format the tables. The writing of the book was partially supported by the Colorado Department of Higher Education as part of the proposal "OER for the Creation of Interactive Computational Notebooks and a Computational Pathway in Mathematics and Statistics".

Chapter 1

R Foundations

Meaningful data analysis requires the use of computer software.

R statistical software is one of the most popular tools for data analysis in academia, industry, and government. In what follows, I will attempt to lay a foundation of basic knowledge and skills with R that you will need for data analysis. I make no attempt to be exhaustive, and many other important aspects of using R (like plotting) will be discussed later, as needed.

1.1 Setting up R and RStudio Desktop

What is R?

R is a programming language and environment designed for statistical computing. It was introduced by Robert Gentleman and Robert Ihaka in 1993 as a free implementation of the *S* programming language developed at Bell Laboratories (https://www.r-project.org/about.html)

Some important facts about R are that:

- R is free, open source, and runs on many different types of computers (Windows, Mac, Linux, and others).
- R is an interactive programming language.
  - You type and run a command in the Console for immediate feedback, in contrast to a compiled programming language, which compiles a program that is then executed.
- R is highly extendable.
  - Many user-created packages are available to extend the functionality beyond what is installed by default.
  - Users can write their own functions and easily add software libraries to R.

Installing R

To install R on your personal computer, you will need to download an installer program from the R Project's website (https://www.r-project.org/). Links to download the installer program for your operating system *should* be found at https://cloud.r-project.org/. Click on the download link appropriate for your computer's operating system and install R on your computer. If you have a Windows computer, a stable link for the most current installer program is available at https://cloud.r-project.org/bin/windows/base/release.html. (Similar links are not currently available for Mac and Linux computers.)

Installing RStudio

RStudio Desktop is a free "front end" for R provided by Posit Software (https://posit.co/). RStudio Desktop makes doing data analysis with R much easier by adding an Integrated Development Environment (IDE) and providing many other features. Currently, you may download RStudio at https://posit.co/download/rstudio-desktop/. You may need to navigate the RStudio website directly if this link no longer functions. Download the Free version of RStudio Desktop appropriate for your computer and install it.

Having installed both R and RStudio Desktop, you will want to open RStudio Desktop as you continue to learn about R.

RStudio Layout

RStudio Desktop has four panes:

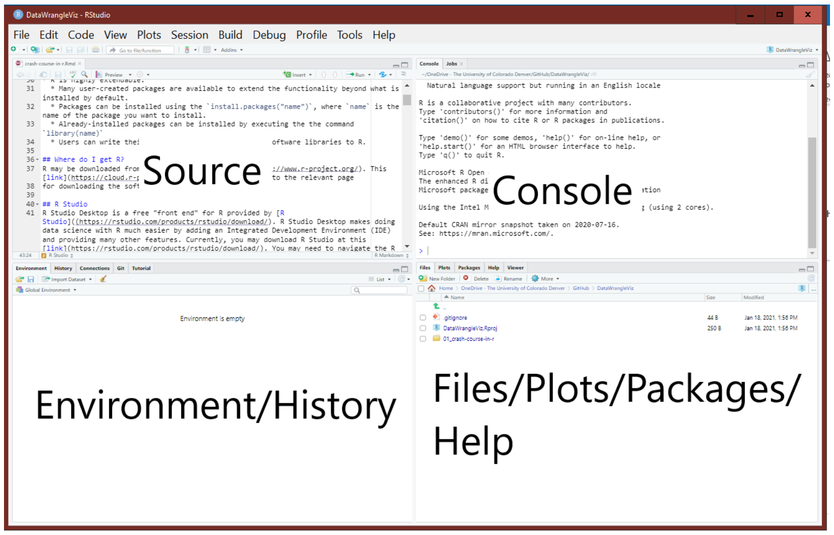
1. Console: the pane where commands are run.
2. Source: the pane where you prepare commands to be run.
3. Environment/History: the pane where you can see all the objects in your workspace, your command history, and other information.
4. The Files/Plot/Packages/Help: the pane where you navigate between directories, where plots can be viewed, where you can see the packages available to be loaded, and where you can get help.

To see all RStudio panes, press the keys `Ctrl + Alt + Shift + 0` on a PC or `Cmd + Option + Shift + 0` on a Mac.

Figure **??** displays a labeled graphic of the panes. Your panes are likely in a different order than the graphic shown because I have customized my workspace for my own needs.

Customizing the RStudio workspace

At this point, I would highly encourage you to make one small workspace customization that will likely save you from experiencing future frustration. R provides a "feature" of that allows you to "save a workspace". This allows you to easily pick up where you left off your last analysis. The issue with this is that over time you accumulate a lot of environmental artifacts that can conflict with each other. This can lead to errors and incorrect results that you will need to



```
  * Many user-created packages are available to extend the functionality beyond what is
installed by default.
  * Packages can be installed using the `install.packages("name")`, where `name` is the
name of the package you want to install.
  * Already-installed packages can be installed by executing the the command
`library(name)`
  * Users can write thei                         oftware libraries to R.

## Where do I get R?
R may be downloaded fron                         ://www.r-project.org/). This
[link](https://cloud.r-p                         to the relevant page
for downloading the soft

## R Studio
R Studio Desktop is a free "front end" for R provided by [R
Studio]((https://rstudio.com/products/rstudio/download/). R Studio Desktop makes doing
data science with R much easier by adding an Integrated Development Environment (IDE)
and providing many other features. Currently, you may download R Studio at this
[link](https://rstudio.com/products/rstudio/download/). You may need to navigate the R
```

```
~/OneDrive - The University of Colorado Denver/GitHub/DataWrangleViz/
  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

Microsoft R Open
The enhanced R di
Microsoft package                        ation

Using the Intel M                        g (using 2 cores).

Default CRAN mirror snapshot taken on 2020-07-16.
See: https://mran.microsoft.com/.

>
```

Home > OneDrive - The University of Colorado Denver > GitHub > DataWrangleViz

Name	Size	Modified
..		
.gitignore	44 B	Jan 18, 2021, 1:56 PM
DataWrangleViz.Rproj	250 B	Jan 18, 2021, 1:56 PM
01_crash-course-in-r		

Figure 1.1: The RStudio panes labeled for convenience.

deal with. Additionally, this "feature" hinders the ability of others to reproduce your analysis because other users are unlikely to have the same workspace.

To turn off this feature, in the RStudio menu bar click Tools → Global Options and then make sure the "General" option is selected. Then make the following changes (if necessary):

1. Uncheck the box for "Restore .RData into workspace at startup".
2. Change the toggle box for "Save workspace to .RData on exit" to "Never".
3. Click Apply then OK to save the changes.

Figure **??** displays what these options should look like.

1.2 Running code, scripts, and comments

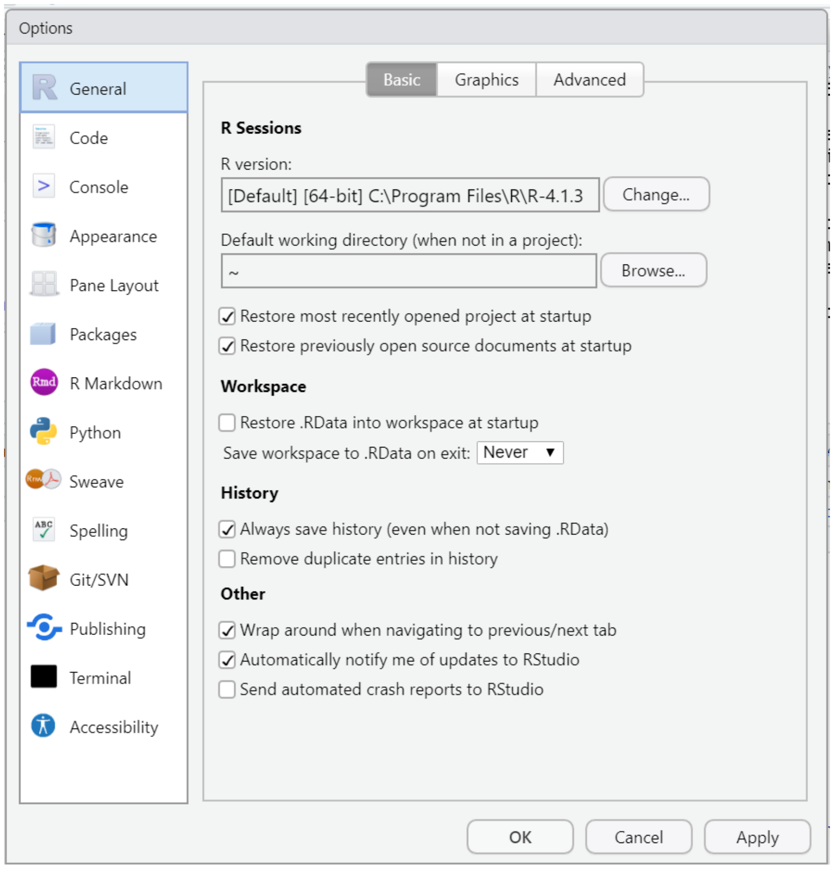
You can run code in R by typing it in the Console next to the > symbol and pressing the Enter key.

If you need to successively run multiple commands, it's better to write your commands in a "script" file and then save the file. The commands in a Script file are often generically referred to as "code".

Script files make it easy to:

- Reproduce your data analysis without retyping all your commands.
- Share your code with others.

A new Script file can be obtained by:



R Sessions

R version: [Default] [64-bit] C:\Program Files\R\R-4.1.3

Default working directory (when not in a project): ~

☑ Restore most recently opened project at startup
☑ Restore previously open source documents at startup

Workspace

☐ Restore .RData into workspace at startup
Save workspace to .RData on exit: Never

History

☑ Always save history (even when not saving .RData)
☐ Remove duplicate entries in history

Other

☑ Wrap around when navigating to previous/next tab
☑ Automatically notify me of updates to RStudio
☐ Send automated crash reports to RStudio

Figure 1.2: The General options window.

- Clicking File → New File → R Script in the RStudio menu bar.
- Pressing `Ctrl + Shift + n` on a PC or `Cmd + Shift + n` on a Mac.

There are various ways to run code from a Script file. The most common ones are:

1. Highlight the code you want to run and click the Run button

at the top of the Script pane.
2. Highlight the code you want to run and press "Ctrl + Enter" on your keyboard. If you don't highlight anything, by default, RStudio runs the command the cursor currently lies on.

To save a Script file:

- Click File → Save in the RStudio menu bar.
- Press `Ctrl + s` on a PC or `Cmd + s` on a Mac.

A comment is a set of text ignored by R when submitted to the Console.

A comment is indicated by the `#` symbol. Nothing to the right of the `#` is executed by the Console.

To comment (or uncomment) multiple lines of code in the Source pane of RStudio, highlight the code you want to comment and press `Ctrl + Shift + c` on a PC or `Cmd + Shift + c` on a Mac.

Your turn
Perform the following tasks:

1. Type `1+1` in the Console and press Enter.
2. Open a new Script in RStudio.
3. Type `mean(1:3)` in your Script file.
4. Type `# mean(1:3)` in your Script file.
5. Run the commands from the Script using an approach mentioned above.
6. Save your Script file.
7. Use the keyboard shortcut to "comment out" some of the lines of your Script file.

1.3 Assignment

R works on various types of objects that we'll learn more about later.

To store an object in the computer's memory we must assign it a name using the assignment operator `<-` or the equal sign `=`.

Some comments:

- In general, both `<-` and `=` can be used for assignment.

- Pressing `Alt + -` on a PC or `Option + -` on a Mac will insert `<-` into the R Console and Script files.
  - If you are creating an R Markdown file, then this shortcut will only insert `<-` if you are in an R code block.
- `<-` and `=` are NOT synonyms, but can be used identically most of the time.

It is best to use `<-` for assigning a name to an object and reserving `=` for specifying function arguments. See Section **??** for an explanation.

Once an object has been assigned a name, it can be printed by running the name of the object in the Console or using the `print` function.

Your turn
Run the following commands in the Console:

```
# compute the mean of 1, 2, ..., 10 and assign the name m
m <- mean(1:10)
m # print m
print(m) # print m a different way
```

After the comment, we compute the sample mean of the values $1, 2, \ldots, 10$, then assign it the name `m`. The next two lines are different mechanisms for printing the information contained in the object `m` (which is just the number 5.5).

1.4 Functions

A function is an object that performs a certain action or set of actions based on objects it receives from its arguments. We use a sequence of function calls to perform data analysis.

To use a function, you type the function's name in the Console (or Script) and then supply the function's "arguments" between parentheses, `()`.

The arguments of a function are pieces of data or information the function needs to perform the requested task (i.e., the function "inputs"). Each argument you supply is separated by a comma, `,`. Some functions have default values for certain arguments and do not need to specified unless something beside the default behavior is desired.

e.g., the `mean` function computes the sample mean of an R object `x`. (How do I know? Because I looked at the documentation for the function by running `?mean` in the Console. We'll talk more about getting help with R shortly.) The `mean` function also has a `trim` argument that indicates the, "… fraction … of observations to be trimmed from each end of `x` before the mean is computed" (R Core Team (2023), `?mean`).