Installing R and RStudio

Ashley I Naimi

Oct 2022

Contents

1 Installing R and RStudio

The R programming language has long been a standard for deploying statistical methods in data. It is one of the most commonly used statistical programming languages, and is available for use under the GNU General Public License. What this means is that everyone has the right to freely run the software, copy it, distribute it, and modify it.

R was developed in the early 1990s by Ross Ihaka and Robert Gentleman, and a beta version released in June of 1995. The official version 1.0.0 was publicly released in 2000.

Since then, R has become an extremely popular, and often misunderstood, programming language for statistics. Misunderstandings often arise as the result of the original intent of the programming language,[1] which was to cater to both the R *user* and the R *developer* or *programmer* (Peng, 2022).

[1] The pre-cursor programming language, in fact: S or S Plus. For details, refer to Peng (2022), Chapter 2

The distinction here between a *user* and *programmer* (though somewhat artificial) is an important one. It's one thing to employ the function in R that can be used to fit a generalized linear model. It's another to write a series of R programs that allow you to input dirty data at one end, and obtain a set of tables, figures, and scientific results ready for publication at the other end.

One consequence of this initial focus on both the user and the developer is that there are usually many many many (yes, that many) ways to accomplish the same task in R. One classic example is subsetting a dataset. For example:

```
set.seed(123)
my_dataset <- data.frame(a = runif(10), b = rnorm(10),
    c = rbinom(10, 1, 0.5))

my_dataset
```

```
##            a          b c
## 1  0.2875775  1.7150650 1
## 2  0.7883051  0.4609162 1
## 3  0.4089769 -1.2650612 1
## 4  0.8830174 -0.6868529 1
## 5  0.9404673 -0.4456620 0
## 6  0.0455565  1.2240818 0
```

```
## 7  0.5281055  0.3598138 1
## 8  0.8924190  0.4007715 0
## 9  0.5514350  0.1106827 0
## 10 0.4566147 -0.5558411 0
```

Let's select the first 4 observations:

```
# using base R
my_dataset[1:4, ]
```

```
##           a          b c
## 1 0.2875775  1.7150650 1
## 2 0.7883051  0.4609162 1
## 3 0.4089769 -1.2650612 1
## 4 0.8830174 -0.6868529 1
```

```
# using base R take 2
index <- 1:4
my_dataset[index, ]
```

```
##           a          b c
## 1 0.2875775  1.7150650 1
## 2 0.7883051  0.4609162 1
## 3 0.4089769 -1.2650612 1
## 4 0.8830174 -0.6868529 1
```

```
# using tidyverse
my_dataset %>%
    slice(1:4)
```

```
##           a          b c
## 1 0.2875775  1.7150650 1
## 2 0.7883051  0.4609162 1
## 3 0.4089769 -1.2650612 1
## 4 0.8830174 -0.6868529 1
```

```
# using head
head(my_dataset, 4)
```

```
##           a          b c
## 1 0.2875775  1.7150650 1
## 2 0.7883051  0.4609162 1
## 3 0.4089769 -1.2650612 1
## 4 0.8830174 -0.6868529 1
```

This is just one simple example of the redundancies in R that often lead to a steep learning curve on how to accomplish task in R. This learning curve can be a real obstacle.[2] Keep in mind, when dealing with the R learning curve, it is sometimes useful to remember the distinction between using and programming R. Here, we focus on elements of both using and programming with a specific focus in mind: generating a suite of results from an input dataset that answers our research questions of interest.

[2] See, e.g.: https://www.statmethods.net/about/learningcurve.html

1.1 What is R?

R is an environment for statistical computing and graphics. It is a relatively simple programming language that allows us to engage engage in statistics using data.

There are many programming languages. For example, C/C++ is a well known **compiled** programming language that requires a complete program to run. The benefit of using C/C++ is that there is no faster alternative.

On the other hand, R is an **interpreted** language. Because of this, commands can be run interactively (i.e., we don't need a complete program to run commands in R). This offers a degree of flexibility, but comes at the cost of speed.

So if C/C++ (or Julia, or Fortran, or other) is faster, why use R? There are several reasons. First, R has over 15,000 function libraries that can be used to deploy a wide range of statistical analysis tools. It implements many common statistical, data science, and machine learning procedures, and it possesses *exceptional* graphics functions.

1.2 How does R compare to SAS/Stata?

Learning R requires a fairly similar commitment compared to SAS/Stata for simple tasks such as basic regression or simple exploratory data analysis. R is much better at plotting and visualizing data, and has a number of different functions available to do so (including base R functions, and well as the more sophisticated `ggplot` functions).

R happens to be much easier to use for complex tasks, particularly when using two-stage estimators, which require fitting a model to the data, and then using information from that model in a second model (second stage).

Overall, I believe that R offers a much greater return on investment than SAS or Stata. For certain tasks, such as machine learning, the repertoire of tools available in R users is **unparalleled**.

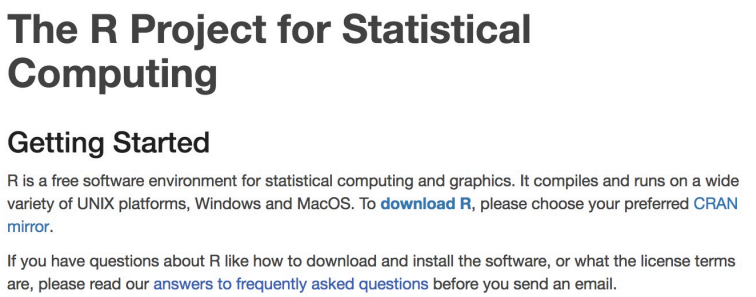
1.3 Installing R

There are several ways to install R. One appraoch would be to install it using a package manager system, such as APT (e.g., `apt-get`) in unix, Homebrew in Mac OS,[3] or Chocolatey in Windows.[4] While these approaches have their benefits, they can add complexity to the install, such as having to specify paths for executables, binaries, and other important library files needed to run R.

[3] https://brew.sh/
[4] https://chocolatey.org/

For this short course, it is highly recommended that you avoid using package managers to install R. Instead, visit the R project website:

www.r-project.org



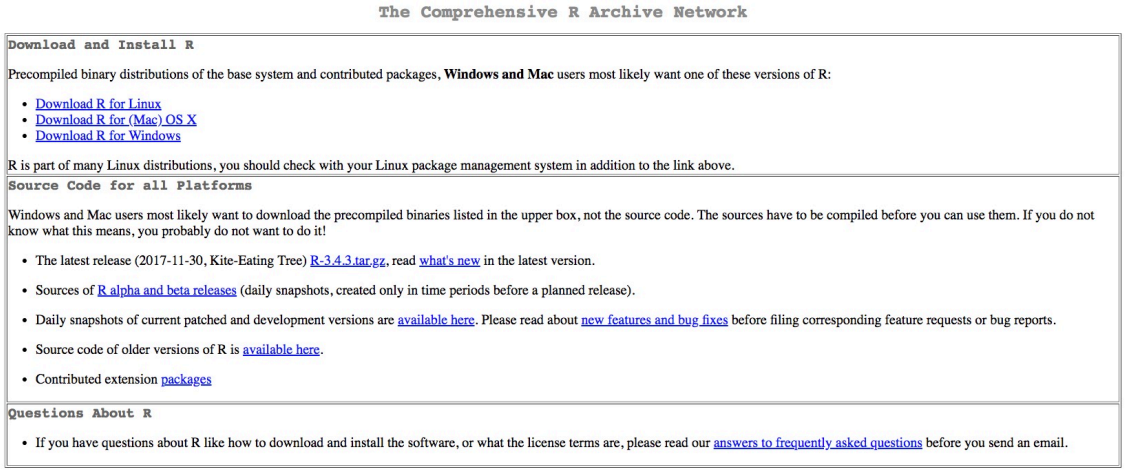
On this site, click the "download R" link, which will take you to a page that includes a list of all the CRAN mirrors.[5]

You should select the mirror that is geographically close to you. I often select the CMU mirror, or the RStudio mirror:

[5] CRAN, or the Comprehensive R Archive Network, is a network of file transfer protocol and website servers that contain an archive of all of the current and all previous versions of R, as well as most of the R packages that you will need to use to deploy the software in your projects.

http://lib.stat.cmu.edu/R/CRAN/

https://cloud.r-project.org/



The Comprehensive R Archive Network

Download and Install R

Precompiled binary distributions of the base system and contributed packages, **Windows and Mac** users most likely want one of these versions of R:

- Download R for Linux
- Download R for (Mac) OS X
- Download R for Windows

R is part of many Linux distributions, you should check with your Linux package management system in addition to the link above.

Source Code for all Platforms

Windows and Mac users most likely want to download the precompiled binaries listed in the upper box, not the source code. The sources have to be compiled before you can use them. If you do not know what this means, you probably do not want to do it!

- The latest release (2017-11-30, Kite-Eating Tree) R-3.4.3.tar.gz, read what's new in the latest version.
- Sources of R alpha and beta releases (daily snapshots, created only in time periods before a planned release).
- Daily snapshots of current patched and development versions are available here. Please read about new features and bug fixes before filing corresponding feature requests or bug reports.
- Source code of older versions of R is available here.
- Contributed extension packages

Questions About R

- If you have questions about R like how to download and install the software, or what the license terms are, please read our answers to frequently asked questions before you send an email.

Click one of the three links at the top of the page that corresponds to your operating system. Once there, you should be able to download a package or executive file that will install the latest release of R on your system.

Once you've downloaded the latest .pkg or .exe file, installing R is similar to installing other software programs. Of note, there are usually several options that you can modify as you install. *For the purpose of this course, you should install R under the default options.*

Once R is successfully installed, the next step is to install RStudio.

1.4 What is RStudio?

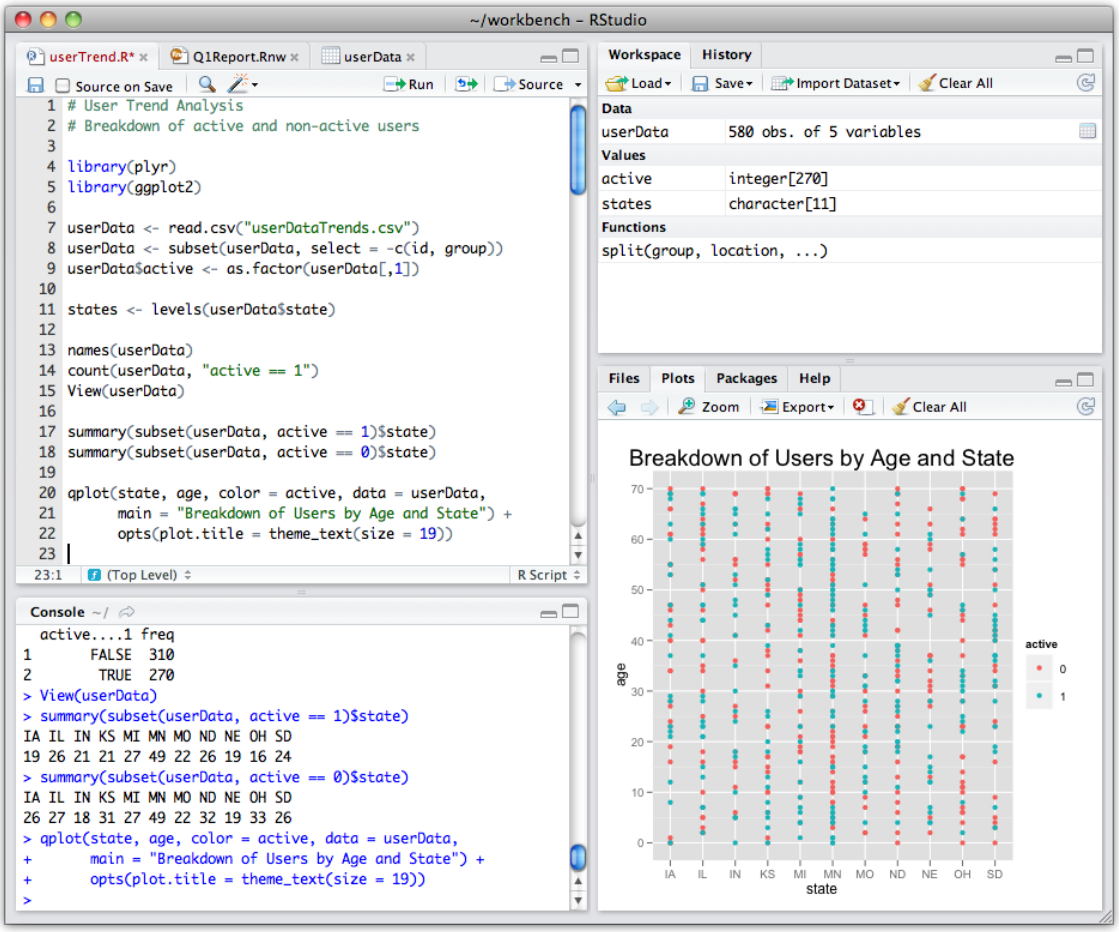
RStudio is an *integrated development environment* (IDE) that provides an interface for you to use R. Basically, an IDE is a software program that facilitates the process of writing, executing, and debugging code. There are many IDEs available, and they are usually not program specific.[6] Some are listed in the following table:

[6] For example, you can use RStudio with both R and Python.

IDE	Website
Rstudio	https://www.rstudio.com/
Atom	https://atom.io/
Sublime Text	https://www.sublimetext.com/

IDE	Website
Notepad++	https://notepad-plus-plus.org/
vim	https://www.vim.org/
emacs	https://www.gnu.org/software/emacs/
micro	https://micro-editor.github.io/
VSCode	https://code.visualstudio.com/

The following figure shows the basic layout of the RStudio IDE. The program interface is divided into four quadrants: the source code panel (usually a .R file or .RMD file, for example); the console, which shows the interactive history of what's being run (this would be the place that commands are sent to R);



the workspace, which shows what is currently loaded in the R environment; and a fourth panel which includes a file/folder directory structure of the current

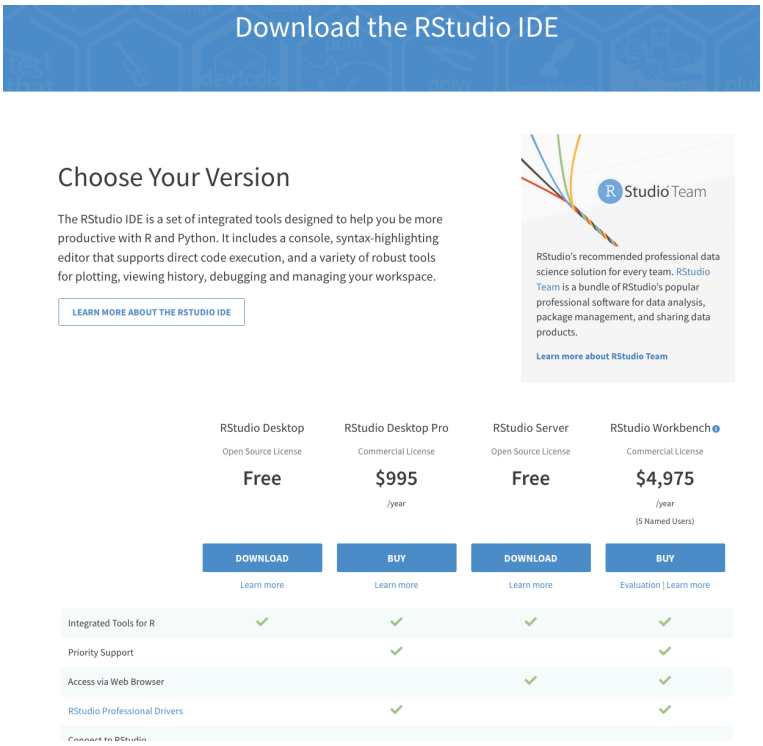
working environment, the plots generated, help files, loaded packages, and other item.[7]

[7] Note that the layout and content of the four panes can be modified through the RStudio options.

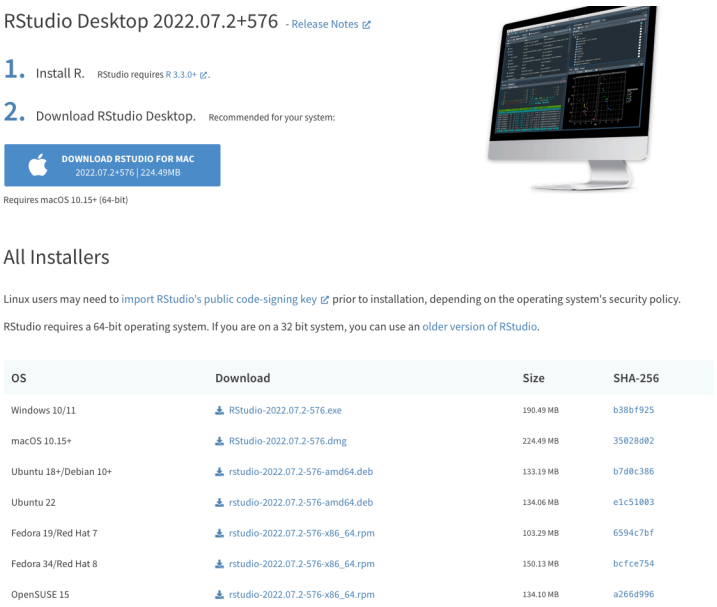
1.5 Installing RStudio

Installing RStudio is also straightforward. The installation files are located on the RStudio Website:

https://www.rstudio.com/products/rstudio/download/



We will be using the freely available Desktop Version of RStudio. Clicking on the download link brings us to a page where we can download the .pkg or the .exe file:



1.6 Setting Up RStudio

Once both R and RStudio are installed, it is important to set up RStudio in such a way that maximizes the efficiency of project development.

Over the course of this session, we will see that there are some important "rules" to follow when seeking to develop code and analyses for a project. Consider, for example, the situation where you spend the day working to generate regression model output, tables, figures, and other elements of the analysis. After a day's work, you will likely have to shut down the program, close your computer, and come back to the analysis session some other time.

The key question then becomes "what do you save from your day's work?" Another way to think of this question is to ask: "what constitutes the main product of your analysis?"

Should it be all the regression model outputs, tables, figures, and other elements of the workspace?

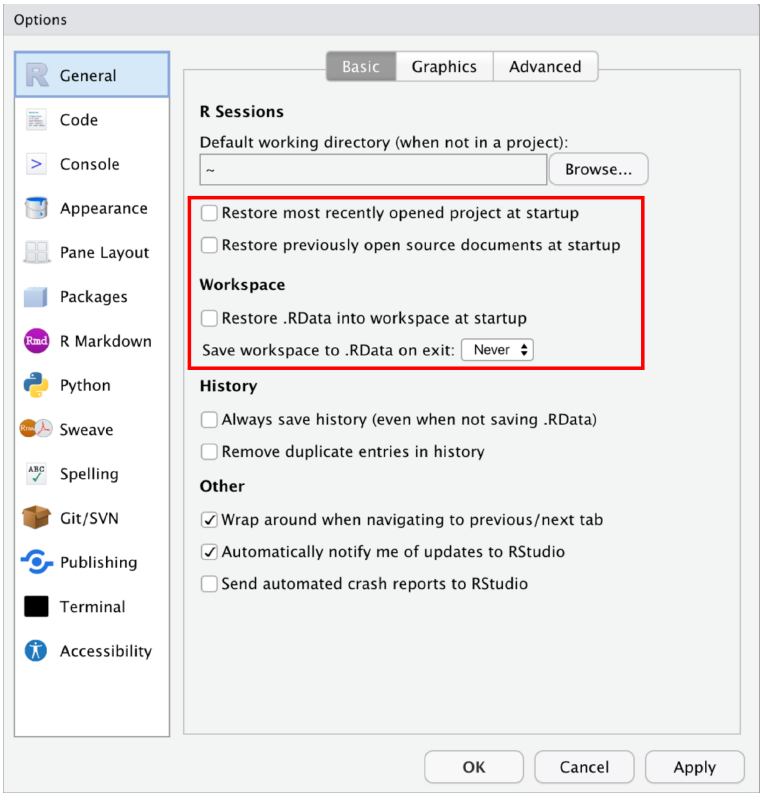
According to modern data science principles, the answer is "no" (see Wickham and Grolemund, 2017, Chapter 8). Rather than focusing on the elements of the analysis that are created in the workspace, the focus should be on the source code that generated those elements. This focus leads to an improved *data science pipeline*, and facilitates the generation of reproducible scientific results.

To set up our R and RStudio programming environment that better aligns with these modern data science principles, there are a few options in the RStudio program that we should change.

These changes should occur in the RStudio options pane:

Windows	**Tools > Options**
Mac	**RStudio > Preferences**

In the pane that arises, you should change the options so that they resemble the options I've selected and displayed in the red box:



Setting these options on your computer as they are in the image above will facilitate the development of a reproducible data science pipeline.

(Take some time to complete exercise 1)

The Data Science Pipeline:

Consider the figure below (adapted from Roger Peng's Lecture on the "Data Science Pipeline": https://bit.ly/3DEx62U) that shows the general process from raw data to scientific results. The objective in creating a "reproducible pipeline" is to simplify the analytic process so that one can execute a few simple commands to re-create original work.

For example, we might write a shell script (.sh file) that runs three programs in R: the processing code, the analytic code, and the presentation code (all .R files). Collectively, these files carry out all the analyses we need to generate the results we'd like to present.

One key consideration in this data science pipeline framework is that the focus is on the source code, and less so on the output from this code. For example, in fitting a generalized linear model, we would not be particularly interested in the fit of this model, but rather the code that fits this model to the data, the code that generates the analytic data, and the code that takes the model output, and generates tables, figures, and other results for reporting to the scientific community. This, in essence, is one of the basic ideas behind the data science pipeline.

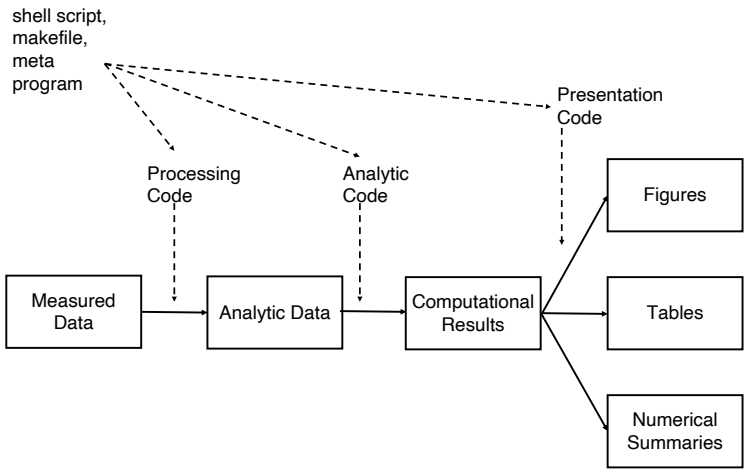


Figure 1: The Data Science Pipeline

References

Roger D Peng. *R Programming for Data Science*. Leanpub, https://bookdown.org/rdpeng/rprogdatascience/, 2022.

Hadley Wickham and Garrett Grolemund. *R for Data Science*. O'Reilly, https://r4ds.had.co.nz/index.html, 2017.