R Is Hard: A Brief Introduction to Using R

Mikaela Westhoff

Draft Edited 8/24/2018

Hello everyone, and welcome to the wonderful world of **R** code writing. I know that starting out with statistical software can be an exercise in patience, but I assure you that you are more than capable of figuring it out. The key is to *keep trying*.

In this brief note, I will explain some of the basics of dealing with writing **R** code. I won't be going into detail about the different commands you can use or packages you can install (those are very well covered in introductory **R** textbooks or online tutorials). Rather, I will focus on some of the critical skills you need to develop in order to become a confident **R** user. These include:

- How to write code clearly and with a consistent style so others can understand
- How to effectively use search engines to find new packages and answers to your coding problems
- How to find and read package documentation and vignettes so you can teach yourself how to use any package/function you might need
- Common coding mistakes to check for when your code won't run
- Resources for teaching yourself the basics and finding answers to your questions

Style Guide for Writing R Code

Once you have been writing code for a while, you will start to notice that there is a big difference between code in terms of readability. Whether you are looking through your colleagues' code for help or perusing replication files of other scholars' work, it can be frustrating when things look like a jumbled mess. Such "spaghetti" code is a result of inconsistent coding style. Coding style refers to the conventions a person uses to write their code. These include variable name formats, commenting, maximum characters per line, use of spaces, placement of brackets, and other similar aspects.

Using a consistent coding style is important so that others can quickly and clearly understand what your code is doing. This helps with replicability of studies, trouble-shooting problematic code when you are learning, and communicating new coding solutions. I highly encourage you to look through existing style guides and pick one to use consistently. There is no de facto **R**-code style for all applications, so feel free to select whatever you find most readable.

A selection of style guides:

- Google's Style Guide
- Hadley Wickhams's Style Guide
- Bioconductor Style Guide
- Bernd Bisch's Guidelines
- Jean Fan's Style Guide
- Henrik Bengstsson's R Conventions

Strategies for Finding Out How to Do Something New

Inevitably when you are working on a new statistics project, you will run into something that you do not know how to code in R. This could be how to create a contour plot, how to run a structural model, or even how to add together a simple vector.

When this happens, the first thing that you should do is do a search in R to see if there is a package or command that you can use. To do this, you can use either the `??` command (ex. `??regression`) or the `help.search` command (ex. `help.search("regression")`). Both of these commands do the same thing: search for the word in the **R** help files. This is particularly useful if you know what the thing you want to do is called you need to use (such as logit, regression, mean, etc) but don't know the exact command name or syntax that **R** wants you to use.

If this initial check inside of **R** doesn't work, do a quick search to see if there is a simple solution somewhere out there on the internet. There usually is. Do this search through Google or, even better, rseek. Rseek is a search engine maintained by Sasha Goodman that functions like Google, but only pulls articles that have to do with **R**. This will help you to more quickly find what you are looking for.

When entering a search term to figure out how to do something in **R**, you may not initially succeed at finding the results you want. This usually means that you aren't using the correct words to communicate what you want or that your search is too vague. When this happens, try rephrasing what you are trying to do in different or more precise terms. Go into your stats text book and find out if you are using the correct terminology for the model you are looking for. It may take some trial and error, but unless you are doing something really cutting edge, there is almost always someone else who has asked the same question as you.

How to Figure Out How an R Command/Function Works

Once you find the command that you want to use for a task, you need to figure out what arguments you need to enter to tell it what to do. To pull up the function documentation that show you this, simply type `?` followed by the name of the function into the command line (ex. `?mean` to see the documentation for finding the arithmetic mean). Doing this will pull up the function documentation in the help window of **R** Studio, found in the lower right corner of the interface. This function documentation contains a number of helpful sections. The most important of these are:

- Description: A general statement of what this function is
- Usage: What the function looks like when you use it. Ex. `mean(x, ...)`
- Arguments: A list of all the components of the function that you can use to tell **R** what you want it to do
- Details: Further clarification regarding the function
- Value: An explanation of what this function will produce
- See Also: Related functions or **R** entries that you may want to look at
- Reference: References used by the function's author(s)
- Examples: Example chunks of code using the function

What are Vignettes and How to Find Them

A package vignette in **R** is a long-form documentation file often detailing what a package does, what sort of commands and arguments it accepts, and example chunks of code that give you a good starting point for your own projects. These vignettes go into much more detail than the function documentation I showed you how to find in the previous segment. Think of it this way: function documentation tell you the words you need to say to **R** to get it to run what you want. Vignettes tell you what those words mean. While not all **R** packages have vignettes, many of them do. Vignettes are one of the most useful things that you can read to learn how to use a new package to run the models and functions you need.

To see a list of all available vignettes for the packages you have loaded in **R**, just type `vignette()` into the command line. This will show you what packages and what commands have vignettes available for you to view. This will be displayed such that each package with available vignettes will have a section. Underneath the package heading, the name of the vignette command for that package will be displayed on the the left, followed by the title of vignette.
For example, if I only had the package "Amelia" loaded into **R** and then executed the `vignette()` command, I would get the following output:

```
Vignettes in package ‘Amelia’:

amelia             Amelia II: A Package for Missing Data
```

```
                    (source, pdf)
```
Looking at this output, we see that the name of the package is “Amelia” and the name of the vignette command is “amelia”. In order to pull up the vignette itself for Amelia, we have to reenter the vignette command, only this time specifying the specific vignette we want. This is done by entering:

```
vignette(“amelia”, package = “Amelia”, lib.loc = NULL, all = TRUE)
```
Entering this command and hitting enter will pull up the PDF of the vignette. Read through this for an explanation of how the package works along with snippets of example code.

Common Mistakes and Trouble-Shooting Strategies

So, you have read the vignettes and the function documentation, but your code has still failed. Don't dispair, as there is often a simple solution.

The first thing to do is to look at the error that you are getting. While these error messages are often long and confusing, they hold the key to figuring out where R is getting confused. Noam Ross back in 2015 went into Stack Overflow to analyze the most common error related questions in order to isolate the most common errors. You can take a look at his full analysis here. I will summarize the main takeaways briefly here.

The most common type of error message is the "could not find function" error. These are usually a sign that there is either a typo in the command you typed (like typing `rnorn()` instead of `rnorm()` to generate a random normal distribution) or that the package containing the function you are trying to use is not loaded into **R**. To trouble-shoot these, first double-check that everything is spelled correctly. R is unforgiving when it comes to typos. Once you have confirmed that all the letters and cases are correct, then check that the packages in which your commands are found are installed and loaded into **R**. Code for installing and loading packages is as follows, using Amelia as the example package:
```
# Install the package "Amelia"
install.packages("Amelia")

# Load and attach the package "Amelia"
library("Amelia")
```
The next most common errors according to Ross are:

- "Error in if" errors: These are a result of non-logical data or missing values passed to an "if" conditional statement
- "Error in eval" errors: These are caused by references to objects that do not exist

- "cannot open" errors: These are caused by tell **R** to read a file/data set that doesn't exist or can't be accessed
- "no applicable method" errors: These are caused by using an object-oriented function on a data type it doesn't support
- "subscript out of bounds" errors: These are caused by trying to access an element or dimension or an object that does not exist
- package errors: These are caused by being unable to install, compile or load a package

There are many other types of errors that **R** might throw you. When you find an error that you do not understand, first try running your code line by line to figure out exactly where you are going wrong. Once you isolate the exact problem area, verify that that segment of code is clean and free of typos and missing brackets. If that does not work, next look at the vignette or documentation for that function to see if they have a "common errors" section that might answer your question. If all else fails, copy the error that you are getting into Google or rseek to see if other researchers have had the same problem. Finally, try detaching all the packages you have loaded and only loading the absolutely necessary ones. Sometimes packages just refuse to play nicely with one another.

Good Resources for Teaching Yourself

There are a ton of resources out there for teaching yourself **R**, so many that it can be intimidating. Here are some resources to get you started on answering your questions:

- R-Bloggers: An aggregated collection of **R**-related blog content. This is a great source of information on a variety of **R**-related topics.
- Quick-R: A great **R** introduction site that covers the basics with clear examples and tutorials.
- Stack Overflow: An online developer community where folks post questions. Make sure to search to see if your question has already been answered before posting.
- CRAN: A collection of all the code and documentation for **R** software. If you want documentation, this is where to find it.
- CRAN: R Manual: This is a massive manual for **R** that will cover most of what you need to know.
- The Art of Programming in R by Matloff: A good guide for learning to program.
- Exploratory Data Analysis in R by Pearson: This is a good book aimed towards new R users that introduces how to use R to conduct exploratory data analysis.