

## Chapter 2

# Introduction to R

### 2.1 Getting Started

R is both a programming language and software environment for statistical computing, which is *free* and *open-source*. To get started, you will need to install two pieces of software:

- [R, the actual programming language](#).
  - Chose your operating system, and select the most recent version, 4.1.0.
- [RStudio, an excellent IDE for working with R](#).
  - Note, you must have R installed to use RStudio. RStudio is simply an interface used to interact with R.

The popularity of R is on the rise, and everyday it becomes a better tool for statistical analysis. It even generated this book! (A skill you will learn in this course.) There are many good resources for learning R.

The following few chapters will serve as a whirlwind introduction to R. They are by no means meant to be a complete reference for the R language, but simply an introduction to the basics that we will need along the way. Several of the more important topics will be re-stressed as they are actually needed for analyses.

These introductory R chapters may feel like an overwhelming amount of information. You are not expected to pick up everything the first time through. You should try all of the code from these chapters, then return to them a number of times as you return to the concepts when performing analyses.

R is used both for software development and data analysis. We will operate in a grey area, somewhere between these two tasks. Our main goal will be to analyze

data, but we will also perform programming exercises that help illustrate certain concepts.

RStudio has a large number of useful keyboard shortcuts. A list of these can be found using a keyboard shortcut – the keyboard shortcut to rule them all:

- On Windows: `Alt + Shift + K`
- On Mac: `Option + Shift + K`

The RStudio team has developed a number of “cheatsheets” for working with both R and RStudio. [This particular cheatsheet for “Base” R](#) will summarize many of the concepts in this document. (“Base” R is a name used to differentiate the practice of using built-in R functions, as opposed to using functions from outside packages, in particular, those from the [tidyverse](#). More on this later.)

When programming, it is often a good practice to follow a style guide. (Where do spaces go? Tabs or spaces? Underscores or CamelCase when naming variables?) No style guide is “correct” but it helps to be aware of what others do. The more important thing is to be consistent within your own code.

- [Hadley Wickham Style Guide](#) from [Advanced R](#)
- [Google Style Guide](#)

For this course, our main deviation from these two guides is the use of `=` in place of `<-`. (More on that later.)

## 2.2 Basic Calculations

To get started, we’ll use R like a simple calculator.

### Addition, Subtraction, Multiplication and Division

Math	R	Result
$3 + 2$	<code>3 + 2</code>	5
$3 - 2$	<code>3 - 2</code>	1
$3 \cdot 2$	<code>3 * 2</code>	6
$3/2$	<code>3 / 2</code>	1.5

### Exponents

Math	R	Result
$3^2$	<code>3 ^ 2</code>	9
$2^{(-3)}$	<code>2 ^ (-3)</code>	0.125
$100^{1/2}$	<code>100 ^ (1 / 2)</code>	10
$\sqrt{100}$	<code>sqrt(100)</code>	10

### Mathematical Constants

Math	R	Result
$\pi$	<code>pi</code>	3.1415927
$e$	<code>exp(1)</code>	2.7182818

### Logarithms

Note that we will use `ln` and `log` interchangeably to mean the natural logarithm. There is no `ln()` in R, instead it uses `log()` to mean the natural logarithm.

Math	R	Result
$\log(e)$	<code>log(exp(1))</code>	1
$\log_{10}(1000)$	<code>log10(1000)</code>	3
$\log_2(8)$	<code>log2(8)</code>	3
$\log_4(16)$	<code>log(16, base = 4)</code>	2

### Trigonometry

Math	R	Result
$\sin(\pi/2)$	<code>sin(pi / 2)</code>	1
$\cos(0)$	<code>cos(0)</code>	1

## 2.3 Getting Help

In using R as a calculator, we have seen a number of functions: `sqrt()`, `exp()`, `log()` and `sin()`. To get documentation about a function in R, simply put a question mark in front of the function name and RStudio will display the documentation, for example:

```
?log  
?sin  
?paste  
?lm
```

Frequently one of the most difficult things to do when learning R is asking for help. First, you need to decide to ask for help, then you need to know *how* to ask for help. Your very first line of defense should be to Google your error message or a short description of your issue. (The ability to solve problems using this method is quickly becoming an extremely valuable skill.) If that fails, and it eventually will, you should ask for help. There are a number of things you should include when emailing an instructor, or posting to a help website such as [Stack Exchange](#).

- Describe what you expect the code to do.
- State the end goal you are trying to achieve. (Sometimes what you expect the code to do, is not what you want to actually do.)
- Provide the full text of any errors you have received.
- Provide enough code to recreate the error. Often for the purpose of this course, you could simply email your entire `.R` or `.Rmd` file.
- Sometimes it is also helpful to include a screenshot of your entire RStudio window when the error occurs.

If you follow these steps, you will get your issue resolved much quicker, and possibly learn more in the process. Do not be discouraged by running into errors and difficulties when learning R. (Or any technical skill.) It is simply part of the learning process.

## 2.4 Installing Packages

R comes with a number of built-in functions and datasets, but one of the main strengths of R as an open-source project is its package system. Packages add additional functions and data. Frequently if you want to do something in R, and it is not available by default, there is a good chance that there is a package that will fulfill your needs.

To install a package, use the `install.packages()` function. Think of this as buying a recipe book from the store, bringing it home, and putting it on your shelf.

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

Once a package is installed, it must be loaded into your current R session before being used. Think of this as taking the book off of the shelf and opening it up to read.

```
library(ggplot2)
```

Once you close R, all the packages are closed and put back on the imaginary shelf. The next time you open R, you do not have to install the package again, but you do have to load any packages you intend to use by invoking `library()`.

