Biological Data Science with R

Stephen D. Turner

2018-12-18

Table of contents

Preface

This book was written as a companion to a series of courses introducing the essentials of biological data science with R. While this book was written with the accompanying live instruction in mind, this book can be used as a self-contained self study guide for quickly learning the essentials need to get started with R. The BDSR book and accompanying course introduces methods, tools, and software for reproducibly managing, manipulating, analyzing, and visualizing large-scale biological data using the R statistical computing environment. This book also covers essential statistical analysis, and advanced topics including survival analysis, predictive modeling, forecasting, and text mining.

This is not a *"Tool X"* or *"Software Y"* book. I want you to take away from this book and accompanying course the ability to use an extremely powerful scientific computing environment (R) to do many of the things that you'll do *across study designs and disciplines* – managing, manipulating, visualizing, and analyzing large, sometimes high-dimensional data. Regardless of your specific discipline you'll need the same computational know-how and data literacy to do the same kinds of basic tasks in each. This book might show you how to use specific tools here and there (e.g., DESeq2 for RNA-seq analysis (Love, Huber, and Anders 2014), ggtree for drawing phylogenetic trees (Yu et al. 2017), etc.), but these are not important – you probably won't be using the same specific software or methods 10 years from now, but you'll still use the same underlying data and computational foundation. That is the point of this series – to arm you with a basic foundation, and more importantly, to enable you to figure out how to use *this tool* or *that tool* on your own, when you need to.

This is not a statistics book. There is a short lesson on essential statistics using R in Chapter 7 but this short chapter offers neither a comprehensive background on underlying theory nor in-depth coverage of implementation strategies using R. Some general knowledge of statistics and study design is helpful, but isn't required for going through this book or taking the accompanying course.

There are no prerequisites to this book or the accompanying course. However, each chapter involves lots of hands-on practice coding, and you'll need to download and install required softwar and download required data. See the setup instructions in Appendix A.

Acknowledgements

This book is partially adapted from material we developed for the University of Virginia BIMS8382 graduate course . The material for this course was adapted from and/or inspired by Jenny Bryan's STAT545 course at UBC (Bryan 2019), Software Carpentry (Wilson 2014) and Data Carpentry (Teal et al. 2015) courses, David Robinson's *Variance Explained* blog (Robinson 2015), the ggtree vignettes (Yu 2022) *Tidy Text Mining with R* (Silge and Robinson 2017), and likely many others.

Part I

Core lessons

1 Basics

This chapter introduces the R environment and some of the most basic functionality aspects of R that are used through the remainder of the book. This section assumes little to no experience with statistical computing with R. This chapter introduces the very basic functionality in R, including variables, functions, and importing/inspecting data frames (tibbles).

1.1 RStudio

Let's start by learning about RStudio. **R** is the underlying statistical computing environment. **RStudio** is a graphical integrated development environment (IDE) that makes using R much easier.

- **Options:** First, let's change a few options. We'll only have to do this once. Under *Tools... Global Options...*:
    - Under *General*: Uncheck "Restore most recently opened project at startup"
    - Under *General*: Uncheck "Restore .RData into workspace at startup"
    - Under *General*: Set "Save workspace to .RData on exit:" to Never.
    - Under *General*: Set "Save workspace to .RData on exit:" to Never.
    - Under *R Markdown*: Uncheck "Show output inline for all R Markdown documents"
- Projects: first, start a new project in a new folder somewhere easy to remember. When we start reading in data it'll be important that the *code and the data are in the same place.* Creating a project creates an Rproj file that opens R running *in that folder.* This way, when you want to read in dataset *whatever.txt*, you just tell it the filename rather than a full path. This is critical for reproducibility, and we'll talk about that more later.
- Code that you type into the console is code that R executes. From here forward we will use the editor window to write a script that we can save to a file and run it again whenever we want to. We usually give it a `.R` extension, but it's just a plain text file. If you want to send commands from your editor to the console, use `CMD+Enter` (`Ctrl+Enter` on Windows).
- Anything after a `#` sign is a comment. Use them liberally to *comment your code.*

1.2 Basic operations

R can be used as a glorified calculator. Try typing this in directly into the console. Make sure you're typing into into the editor, not the console, and save your script. Use the run button, or press CMD+Enter (Ctrl+Enter on Windows).

```
2+2
5*4
2^3
```

R Knows order of operations and scientific notation.

```
2+3*4/(5+3)*15/2^2+3*4^2
5e4
```

However, to do useful and interesting things, we need to assign *values* to *objects*. To create objects, we need to give it a name followed by the assignment operator `<-` and the value we want to give it:

```
weight_kg <- 55
```

`<-` is the assignment operator. Assigns values on the right to objects on the left, it is like an arrow that points from the value to the object. Mostly similar to `=` but not always. Learn to use `<-` as it is good programming practice. Using `=` in place of `<-` can lead to issues down the line. The keyboard shortcut for inserting the `<-` operator is `Alt-dash`.

Objects can be given any name such as `x`, `current_temperature`, or `subject_id`. You want your object names to be explicit and not too long. They cannot start with a number (`2x` is not valid but `x2` is). R is case sensitive (e.g., `weight_kg` is different from `Weight_kg`). There are some names that cannot be used because they represent the names of fundamental functions in R (e.g., `if`, `else`, `for`, see here for a complete list). In general, even if it's allowed, it's best to not use other function names, which we'll get into shortly (e.g., `c`, `T`, `mean`, `data`, `df`, `weights`). In doubt check the help to see if the name is already in use. It's also best to avoid dots (`.`) within a variable name as in `my.dataset`. It is also recommended to use nouns for variable names, and verbs for function names.

When assigning a value to an object, R does not print anything. You can force to print the value by typing the name:

```
weight_kg
```

Now that R has `weight_kg` in memory, we can do arithmetic with it. For instance, we may want to convert this weight in pounds (weight in pounds is 2.2 times the weight in kg).

```
2.2 * weight_kg
```

We can also change a variable's value by assigning it a new one:

```
weight_kg <- 57.5
2.2 * weight_kg
```

This means that assigning a value to one variable does not change the values of other variables. For example, let's store the animal's weight in pounds in a variable.

```
weight_lb <- 2.2 * weight_kg
```

and then change `weight_kg` to 100.

```
weight_kg <- 100
```

What do you think is the current content of the object `weight_lb`? 126.5 or 220?

You can see what objects (variables) are stored by viewing the Environment tab in Rstudio. You can also use the `ls()` function. You can remove objects (variables) with the `rm()` function. You can do this one at a time or remove several objects at once. You can also use the little broom button in your environment pane to remove everything from your environment.

```
ls()
rm(weight_lb, weight_kg)
ls()
weight_lb # oops! you should get an error because weight_lb no longer exists!
```

Exercise 1

What are the values after each statement in the following?

```
mass <- 50              # mass?
age  <- 30              # age?
mass <- mass * 2        # mass?
age  <- age - 10        # age?
mass_index <- mass/age  # massIndex?
```

1.3 Functions

R has built-in functions.

```r
# Notice that this is a comment.
# Anything behind a # is "commented out" and is not run.
sqrt(144)
log(1000)
```

Get help by typing a question mark in front of the function's name, or `help(functionname)`:

```r
help(log)
?log
```

Note syntax highlighting when typing this into the editor. Also note how we pass *arguments* to functions. The `base=` part inside the parentheses is called an argument, and most functions use arguments. Arguments modify the behavior of the function. Functions some input (e.g., some data, an object) and other options to change what the function will return, or how to treat the data provided. Finally, see how you can *next* one function inside of another (here taking the square root of the log-base-10 of 1000).

```r
log(1000)
log(1000, base=10)
log(1000, 10)
sqrt(log(1000, base=10))
```

> Exercise 2
>
> See `?abs` and calculate the square root of the log-base-10 of the absolute value of `-4*(2550-50)`. Answer should be `2`.

1.4 Tibbles (data frames)

There are *lots* of different basic data structures in R. If you take any kind of longer introduction to R you'll probably learn about arrays, lists, matrices, etc. We are going to skip straight to the data structure you'll probably use most – the **tibble** (also known as the data frame). We use tibbles to store heterogeneous tabular data in R: tabular, meaning that individuals or observations are typically represented in rows, while variables or features are represented as columns; heterogeneous, meaning that columns/features/variables can be different classes (on variable, e.g. age, can be numeric, while another, e.g., cause of death, can be text).

We'll learn more about tibbles in Chapter 2.

2 Tibbles

Not much to see here...

3 Data Manipulation

Not much to see here...

4 Data Visualization

Not much to see here...

5 Tidy EDA

Not much to see here...

6 R Markdown

Not much to see here...

Part II

Electives

7 Essential Statistics

Not much to see here...

8 Survival Analysis

Not much to see here...

9 Predictive Modeling

Not much to see here...

10 Probabilistic Forecasting

Not much to see here...

11 Text Mining

Not much to see here...

12 Phylogenetic Trees

Not much to see here...

13 RNA-seq

Not much to see here...

Summary

In summary, this book has no content whatsoever.

References

Bryan, Jennifer. 2019. "STAT 545: Data Wrangling, Exploration, and Analysis with r." https://stat545.com/.

Love, Michael I., Wolfgang Huber, and Simon Anders. 2014. "Moderated Estimation of Fold Change and Dispersion for RNA-seq Data with DESeq2." *Genome Biology* 15 (12): 1–21.

Robinson, David. 2015. "Variance Explained." http://varianceexplained.org/.

Silge, Julia, and David Robinson. 2017. *Text Mining with R: A Tidy Approach*. 1st edition. Beijing ; Boston: O'Reilly Media.

Teal, Tracy K., Karen A. Cranston, Hilmar Lapp, Ethan White, Greg Wilson, Karthik Ram, and Aleksandra Pawlik. 2015. "Data Carpentry: Workshops to Increase Data Literacy for Researchers."

Wilson, Greg. 2014. "Software Carpentry: Lessons Learned." *F1000Research* 3.

Yu, Guangchuang. 2022. "Ggtree: An r Package for Visualization of Tree and Annotation Data." http://bioconductor.org/packages/ggtree/.

Yu, Guangchuang, David K. Smith, Huachen Zhu, Yi Guan, and Tommy Tsan-Yuk Lam. 2017. "Ggtree: An R Package for Visualization and Annotation of Phylogenetic Trees with Their Covariates and Other Associated Data." *Methods in Ecology and Evolution* 8 (1): 28–36.

A Setup

A.1 Software

A.2 Data

1. **Option 1: Download all the data**. Download and extract this zip file (NA Mb) with all the data for the entire workshop. This may include additional datasets that we won't use here.
2. **Option 2: Download individual datasets as needed.**

   - Create a new folder somewhere on your computer that's easy to get to (e.g., your Desktop). Name it `bds`. Inside that folder, make a folder called `data`, all lowercase.
   - Download individual data files as needed, saving them to the new `bdsr/data` folder you just made. Click to download. If data displays in your browser, right-click and select *Save link as…* (or similar) to save to the desired location.

-

B Additional Resources

Not much to see here...