Data Analysis for Educational Research in R

Joshua M. Rosenberg

2017-07-30

Contents

Chapter 1

Introduction

Educational research is hard to do (Berliner, 2002). This is because many educational phenomena are part of a complex system, with multiple, nested levels, and, well, people, many of them developing. Data analysis in education reflects some of the challenges of educational research writ large. In short, both educational research and analysis of educational data is hard. The goal of this book is to share how to make these difficulties less challenging using R, the open-source, free programming language and software.

Chapter 2

Why a book on data science in educational research

There are at least three reasons why data analysis in educational research is hard:

- Educational researchers have unique methods: an emphasis on multi-level models, networks, and measurement are just some examples.
- Educational researchers face unique challenges: coming from myriad backgrounds, and working in fields with greater or lesser emphases on different aspects of data analysis.
- Finally, there are training challenges. Educational research features some great methodologists: Many advances in the fields mentioned earlier in this session have been made by those working primarily in educational research. Nevertheless, few quantitative classes teach data analysis.

Chapter 3

Getting started and loading data

What you want to do:

- Install R Studio and R
- Load data saved in a Microsoft Excel spreadsheet (`.xlsx`), comma separated values file (`.csv`), SPSS file (`.sav`), or Google Sheet.

3.1 Install R Studio and R

First, you'll need to download the latest versions of R Studio and R. Although we'll exclusively be using R Studio, R Studio needs to have R installed, as well. You can find links here:

Download R Studio: https://www.rstudio.com/products/rstudio/download/#download

Download R: https://cran.r-project.org/

Don't worry; you won't mess anything up if you download (or even install!) the wrong file. Once you've installed both, we can get to work doing some data analysis.

3.2 Loading data

You might be thinking that an Excel file is the first that we would load, but there happens to be a format which you can open and edit in Excel that is even easier to use between Excel and R and among Excel, R, as well as SPSS and other statistical software, like MPlus, and even other programming languages, like Python. That format is CSV, or a comma-separated-values file.

The CSV file is useful because you can open it with Excel and save Excel files as CSV files. Additionally, and as its name indicates, a CSV file is rows of a spreadsheet with the columns separated by commas, so you can view it in a text editor, like TextEdit for Macintosh, as well. Not surprisingly, Google Sheets easily converts CSV files into a Sheet, and also easily saves Sheets as CSV files.

For these reasons, we start with reading CSV files.

3.2.1 Loading CSV files

The easiest way to read a CSV file is with the function `read_csv()`. It is from a package, or an add-on to R, called `readr`, but we are going to install `readr` as well as other packages that work well together as part of a group of packages named the `tidyverse`. To install all of the packages in the tidyverse, use the following command:

```
install.packages("tidyverse")
```

You can also navigate to the Packages pane, and then click "Install", which will work the same as the line of code above. Note, here there is a way to install a package using code or part of the R Studio interface. Usually, using code is a bit quicker, but sometimes (as we will see in a moment) using the interface can be very useful and sometimes complimentary to use of code.

We have now installed the tidyverse. We only have to install a package once, but to use it, we have to load it each time we start a new R session. We will discuss what an R session is later on; for now, know that we have to load a package to use it. We do that with `library()`:

```
library(tidyverse)
```

And now we can load a file. We are going to call the data `student_responses`:

Since we loaded the data, we now want to look at it. We can just type its name, and a summary of the data will print:

This was a minor task, but if you loaded a file and printed it, give yourself a pat on the back. It is no joke to say that many times simply being able to load a file into new software. We are now well on our way to carrying out analysis of our data.

3.2.2 Loading Excel files

We will now do the same with an Excel file. You might be thinking that you can simply open the file in Excel and then save it as a CSV. This is generally a good idea. At the same time, sometimes you may need to directly read a file from Excel, and it is easy enough to do this.

The package that we use, `readxl`, is not a part of the tidyverse, so we will have to install it first (remember, we only need to do this once), and then load it using `library(readxl)`. Note that the command to install `readxl` is grayed-out below: The `#` symbol before `install.packages("readxl")` indicates that this line should be treated as a comment and not actually run, like the lines of code that are not grayed-out. It is here just as a reminder that the package needs to be installed if it is not already.

Once we have installed readxl, we have to load it (just like tidyverse):

```
# install.packages("readxl")
library(readxl)
```

We can then use `read_xlsx()` in the same way as `read_csv()`:

```
x <- read_xls()
x <- read_xlsx()
```

And we can print the data we loaded in the same way:

3.3 Loading SAV files

The same factors that apply to reading Excel files apply to reading `SAV` files (from SPSS). First, install the package `haven`, load it, and the use the function `read_sav()`:

```
# install.packages("haven")
library(haven)
```

```
x <- read_sav("")
```

3.3.1 Google Sheets

Finally, it can sometimes be useful to load a file directly from Google Sheets, and this can be done using the Google Sheets package.

```
# install.packages("googlesheets")
library(googlesheets)
```

When you run the command below, a link to authenticate with your Google account will open in your browser.

```
my_sheets <- gs_ls()
```

You can then simply use the `gs_title()` function in conjunction with the `gs_read()` function:

```
df <- gs_title('title')
df <- gs_read(df)
```

3.4 Conclusion

Great job! You've made it through the first chapter. For more on reading files, we will discuss how to use functions to read every file in a folder (or, to write many different files to a folder).

Chapter 4

Processing and Tidying Data

4.1 What you want to do

- Create new variables
- Select some cases
- Join data

Chapter 5

Processing and Tidying Data

5.1 What you want to do

- Go from wide form to long form
- Go from long form to wide form
- Aggregate data

Chapter 6

Data Visualization

We have finished a nice book.

Chapter 7

Linear Models (Regression and ANOVA)

Some *significant* applications are demonstrated in this chapter.

Chapter 8

Linear Mixed Effects Models

Chapter 9

Social Network Analysis

Chapter 10

Factor Analysis

Chapter 11

Latent Variable Models

Chapter 12

Natural Language Processing

Chapter 13

Cluster Analysis

Chapter 14

Reproducibility

Bibliography