You Have To Get Your Data From Somewhere

Andy Grogan-Kaylor

2022-10-31

Contents

1 You Have To Get Your Data From Somewhere

In learning R, as well as RMarkdown, one of the most difficult tasks seems to be understanding how to import data.

Read your data into a dataset using the *right function* for the *right format* from the *correct location*.

Note that while learning the correct syntax is *very helpful*, RStudio can generate much of this syntax for you using the **File | Import Dataset | ...** menu.

2 RMarkdown

When you are working with RMarkdown, each RMarkdown document creates its own separate **working environment**. Therefore each RMarkdown document needs to have a line, or lines, of code that import the data file for that individual RMarkdown document. e.g.

If data are in R format ...

```
mydata <- load("/project1/mydata.RData") # load R format data
```

If data are in Excel format ...

```
library(readxl) # library to read Excel

mydata <- read_excel("/project1/mydata.xlsx") # load data from Excel
```

3 Basic Idea

In web versions of this document, hover over the footnotes to see what different parts of this syntax do.

`mydata`[1] `<- function`[2]`("path/to/`[3]`file.extension`[4]`")`[5]

4 Setting Up Your Data Before Importing

Your data may already be set up in this way, but ...

Your data, whether in a text file, a statistical file, or an Excel file, should be in *rectangular* format organized by rows and columns, as seen in the example below.

Remember that any statistical or data visualization software is going to be happier with **shorter column names** like *happy*, rather than longer column names like *"Generally speaking, how happy would you say that you are on most days?"*.

id	happy	income	neighborhood
12	12	12	12
123	123	123	123
1	1	1	1

Datasets can be much larger than the example above, but should most often follow this *rectangular* format.

[1] How R will refer to your data. R's *nickname* for your data.
[2] The correct function for the type of data you are reading in, e.g. RData, CSV, Excel.
[3] Where is your data located? The *directory path* to your data.
[4] What is the *filename* of the file containing your data? Note that the extension often tells you *what kind* of data this is.
[5] Sometimes, especially on a Mac, it is necessary to refer to your directory with a `~`, e.g. `~/downloads/project1.RData` or `~/Desktop/project1.RData` or `~/project1/project1.RData`.

5 Examples

5.1 Data Already in R Format

```
mydata <- load("/project1/mydata.RData") # load R format data
```

5.2 Data in CSV Format

CSV stands for *comma separated values*, and is essentially a raw text format for storing data. CSV is often an excellent format for exchanging data between programs. A few lines of *simulated data on clients* in CSV format are reproduced below.

```
"ID","age","gender","program","mental_health_T1","mental_health_T2","latitude","longitude"
2941,32,"Male","Program A",98.81,95.49,42.1108308238603,-83.6103627437424
2745,22,"Other Identity","Program B",86.28,104.84,42.0016810856589,-83.8064503632259
1320,28,"Female","Program B",89.17,107.48,42.0398163096771,-83.6793088312261
1211,20,"Male","Program D",94.15,95.71,42.2673004816002,-83.8247411126595
2293,20,"Female","Program D",85.38,105.09,42.300730845518,-83.7526918820329
```

```
library(readr) # library to read CSV

mydata <- read_csv("/project1/mydata.csv")
```

5.3 Data in Excel Format

```
library(readxl) # library to read Excel

mydata <- read_excel("/project1/mydata.xlsx")
```

5.4 Data in Stata

```
library(haven) # library to read other file formats

mydata <- read_stata("/project1/mydata.dta")
```

5.5 Data in SPSS Format

```
library(haven) # library to read other file formats

mydata <- read_sav("/project1/mydata.sav")
```

5.6 Data in SAS Format

```
library(haven) # library to read other file formats

mydata <- read_sas("/project1/mydata.sas7bdat")
```

5.7 Hand Code Data Using `tribble`

This can be especially useful for hand coding *small amounts* of data that come from a published online or print data source.

```
library(tibble) # library for updated dataframes
```

```
## Warning: package 'tibble' was built under R version 4.1.2
```

```
mydata <- tribble(
  ~ID, ~age, ~gender, ~program,
2941,32,"Male","Program A",
2745,22,"Other Identity","Program B",
1320,28,"Female","Program B",
2293,20,"Female","Program D"
)
```

When importing data using tribble, it is helpful to get a summary of your data.

```
summary(mydata)
```

```
##        ID            age          gender            program
##  Min.   :1320   Min.   :20.0   Length:4           Length:4
##  1st Qu.:2050   1st Qu.:21.5   Class :character   Class :character
##  Median :2519   Median :25.0   Mode  :character   Mode  :character
##  Mean   :2325   Mean   :25.5
##  3rd Qu.:2794   3rd Qu.:29.0
##  Max.   :2941   Max.   :32.0
```

It may then be helpful to insure some variables are changed to `factors`.

```
mydata$gender <- factor(mydata$gender)

mydata$program <- factor(mydata$program)

summary(mydata)
```

```
##        ID            age                   gender       program
##  Min.   :1320   Min.   :20.0   Female         :2   Program A:1
##  1st Qu.:2050   1st Qu.:21.5   Male           :1   Program B:2
##  Median :2519   Median :25.0   Other Identity:1   Program D:1
##  Mean   :2325   Mean   :25.5
##  3rd Qu.:2794   3rd Qu.:29.0
##  Max.   :2941   Max.   :32.0
```

5.8 Data in R Package

Sometimes data is obtained by calling an R package, or the data is directly built into R.

```
library(palmerpenguins) # call Palmer Penguins library

data("penguins") # call the penguins data
```

```
data(iris) # call the iris flower data built into R
```

6 Save Your Data in R Format

Once your data has been imported into R, you may wish to save it to R format for future use.

```
save(mydata, file = "mydata.RData")
```

Alternatively, one of the advantages of R is that it can read data from so many formats. It may be that your data is consistently being updated by other members of your team. For example, members of your team may be constantly updating an Excel file with your data. In such cases, you may wish to keep a line in your script to always import the most recent version of your data from that other format.

7 Working Directory

R uses the concept of a *working directory* to know where to find files, and where to save files.

If you do not specify a particular path to the data file that you are trying to import, R will assume that it is in your working directory.

It is often helpful to simply set your working directory to a particular location and by default, files will be accessed from, and saved to, that directory e.g.:

```
getwd() # "get", or find out, your working directory

setwd("/project1") # set your working directory
```

- Note that R uses a forward slash `/` to specify directory paths. R does not understand the use of a backward slash `\` to specify directories. R uses `~` to refer to the user's (usually your) home directory.
- If you are working in RStudio, you can use the menu option *Session | Set Working Directory | Choose Directory* to choose a particular working directory.
- If you double click on a `*.Rmd` file to start RStudio, R will assume that your working directory is the directory in which that `*.Rmd` file is located. Thus, it is often a good idea to have your data and RMarkdown document saved in the same directory.