R Workshop - Reading data into R

A Michelle Edwards, Ph.D., MLIS

August 28, 2019

Table of Contents

Entering your data into Excel

Best Practices for data entry

When we start to enter data into Excel or any other program, we tend to enter it in a way that makes the most sense for data entry. We want it to be quick, easy to understand, and efficient. However, for statistical analysis purposes, we need to ensure that our data is entered in a way to makes the most sense for the analysis we are about to conduct. I tell my students that one of the most time consuming parts of any data analysis will be the time it takes to clean your data and get it ready for the analysis.

Here are a few recommendations or best practices to ensure your data is ready to be analyzed. Let's start with the variable names or the column names if you wish:

1. Variable names should be set to a maximum length of 32 characters - even that is VERY long
2. ALWAYS start variable names with a letter
3. Numbers can be used anywhere in the variable name AFTER the first character
4. Do NOT use blanks or spaces
5. Underscores "_" and periods "." may be used in a variable name
6. Use lowercase for your variable name

Please note that R is not as restrictive with both variable names and dataframe names, however, it is good practice to be consistent with whatever naming convention you choose to use.

Knowing your data

Before bringing your data into any statistical package, you should take a look at it - or at least be comfortable with it. Chances are it has been entered using Excel. Open your Excel program and browse the data. Here is a snippet of a sample dataset.

Sample Data Table

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
Mazda RX4	21.0	6	160	110	3.90	2.620	16.46	0	1	4	4
Mazda RX4 Wag	21.0	6	160	110	3.90	2.875	17.02	0	1	4	4
Datsun 710	22.8	4	108	93	3.85	2.320	18.61	1	1	4	1
Hornet 4 Drive	21.4	6	258	110	3.08	3.215	19.44	1	0	3	1
Hornet Sportabout	18.7	8	360	175	3.15	3.440	17.02	0	0	3	2

1. Is your data in the right columns?
2. Are there values that look out of line? Maybe a typo?
3. How many observations do you have? Does it match up to what was collected and entered?
4. Are there missing data? Should there be?

Different ways to read data into R

As with many statistical programs we use today, there are a number of ways to read data into R. We will walk through 2 different ways of doing this here. The first will be reading a CSV file and the second will be reading an Excel file.

1. Reading CSV files

What is a CSV file?

A CSV or Comma Separated Value file, is essentially a text file. A file that can be read using Notepad, TextEdit, or any text program. There is no formatting included in these files, it's just numbers, letters, and commas in the case of a CSV file. You do not need any licensed software to read it. To create a CSV file from Excel use the File - SaveAs option in your Excel program. Note that when you use the CSV option, it will only save one worksheet at a time. This is fine!

*Note: When working with your data in Excel, make the action of saving the CSV version of your file part of your workflow. A copy of your Excel file will be your MASTER datafile, and your CSV file will be a copy of your WORKING datafile. You make changes in Excel, save it and save a second copy as a CSV. Then all changes are made one time. Keep the MASTER Excel file where you make your edits, save it as a CSV, which is the file you read into R to perform all of your statistical analyses. This is all part of an effective and efficient Research Data Management lifecycle.

Read.table function

There are so many options to reading the data into R, but I will start by looking at the read.table function. This is a function that is available with Base R. The read.table function allows you to use the **file.choose()** option - this results in a pop-up file browser dialogue box which allows you to navigate your computer to find the CSV file you want to read.

header = TRUE / FALSE allows you to tell R whether you have a header in your CSV file. In other words, does the first row in your dataset contain the variable names? If it does then you will set the **header=TRUE** and if it does not set **header=FALSE**

sep="," tells R that you have commas separating your values - or that you are trying to read a CSV file. If you are reading a file that has another character separating your values, you would change the comma to that character.

For this example I will read a CSV file from my computer and save it as an object called **workshop.data**

```
#workshop.data <- read.table(file.choose(), header=T, sep=",")
```

If you are not sure where your files are, then the file.choose() option may be a great approach for you. One thing to be aware of though - is that this process uses a pop-up window - when the file dialogue window pops up, it may be hidden by the programs you already have open on your computer. Run the statement once, wait, if you do not see the dialogue box pop-up, then check behind the programs you have currently running - it will be open somewhere.

If using the **file.choose()** option is too cumbersome, then set your working directory and specify the file you are reading in as follows:

```
setwd("~/Workshops/R/F19")

workshop.data <- read.table("January.csv", header=T, sep=",")
```

Viewing the data in R

So - it looks like everything worked fine! But, because we saved our data into an object during our process, there is no listing of the data. How do we ensure we've read it correctly?

Environment Window

We should all see a new entry called workshop.data in the Global Environment window in the top right corner of RStudio. It shows 25 obs of 4 variables. If you click on the blue arrow, you will see the structure of the variables: ID, Trmt, Month, Weight_jan and the format of each variable.

List the data

The environment window is nice to see the structure of the data, but I want to SEE my data! Simply typing the name of the object you would like to see - will produce an output with the contents of that object - in this case the data.

```
workshop.data

##    ID Trmt   Month Weight_jan
## 1   1    a January         21
## 2   2    a January         21
## 3   3    a January         17
## 4   4    a January         24
## 5   5    a January         16
## 6   6    b January         15
## 7   7    b January         16
## 8   8    b January         13
## 9   9    b January         13
## 10 10    b January         14
## 11 11    c January         20
## 12 12    c January         23
## 13 13    c January         13
## 14 14    c January         19
## 15 15    c January         19
## 16 16    d January         15
## 17 17    d January         19
## 18 18    d January         23
## 19 19    d January         21
## 20 20    d January         18
## 21 21    e January         16
## 22 22    e January         17
## 23 23    e January         15
## 24 24    e January         24
## 25 25    e January         13
```

For a larger file, viewing all the entire contents of your data file may not be feasible. Using the head() and tail() functions may be a better option.

```
# To view the top 6rows of the data
head(workshop.data)
```

```
##   ID Trmt   Month Weight_jan
## 1  1    a January         21
## 2  2    a January         21
## 3  3    a January         17
## 4  4    a January         24
## 5  5    a January         16
## 6  6    b January         15
```

```
# To view the last 6 rows of the data
tail(workshop.data)
```

```
##    ID Trmt   Month Weight_jan
## 20 20    d January         18
## 21 21    e January         16
## 22 22    e January         17
## 23 23    e January         15
## 24 24    e January         24
## 25 25    e January         13
```

2. Reading Excel files

The **readxl** package was developed to read a worksheet from an Excel file directly into R. If you are using this for the first time, you will first need to install the package and then run the library as follows:

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

library(readxl)
```

To make our work easy from here on, let's make sure that we set our working directory to a location on our computers

```
setwd("~/Workshops/R/F19")
```

Now let's read the January worksheet in the Workshop Excel file.

```
jan.data <- read_excel("Workshop.xlsx", sheet ="January", col_names = TRUE )
```

What happened?

read_excel is the function in the **readxl** package that will read an Excel file

sheet="January" reads the worksheet called January that is found in the Workshop Excel file. Note the quotes around the worksheet name. Also note that R is case-sensitive!

col_names=TRUE is letting R know that we have column headings - with the **read.table** we used **header=TRUE** - this does the same thing.

Review how to view the data

```
jan.data

## # A tibble: 25 x 4
##        ID Trmt  Month   Weight_jan
##     <dbl> <chr> <chr>        <dbl>
##  1      1 a     January         21
##  2      2 a     January         21
##  3      3 a     January         17
##  4      4 a     January         24
##  5      5 a     January         16
##  6      6 b     January         15
##  7      7 b     January         16
##  8      8 b     January         13
##  9      9 b     January         13
## 10     10 b     January         14
## # ... with 15 more rows
```

Exercise

If you have an Excel file on your laptop that contains some of your own research data, try bringing it into R. Remember to review the Best Practices listed above.

If you do not have an Excel with your own research data, create a dummy file and bring it into R Studio.

Are you confident that you can bring your data into R? If so, see you next time when we talk about Descriptive statistics and Visualizing your data in R.