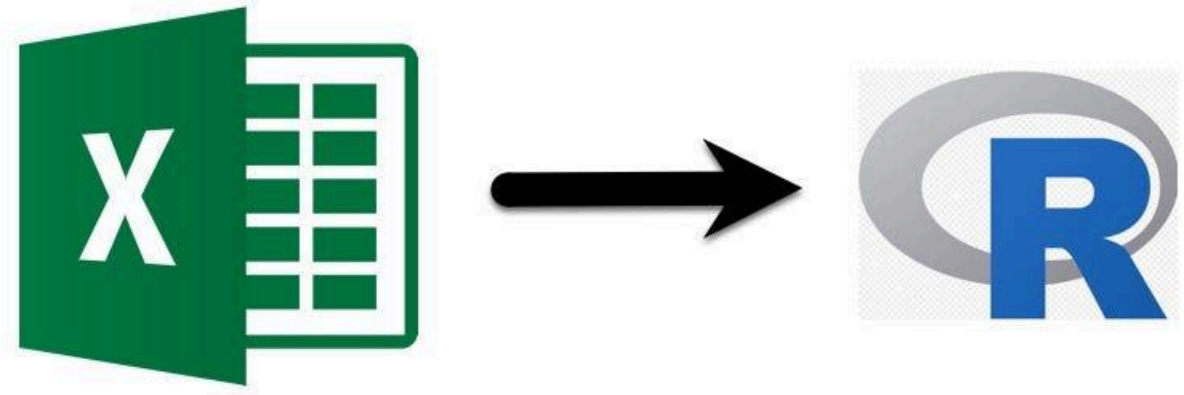
Replace Excel With R

Analysis Needs Reproducibility

By Daran J. Johnson



Introduction

Excel is great for spreadsheet use. R is great for data analysis, reporting, data visualization and modeling. I can remember taking accounting in college, when I was not aware of Excel (it was a long time ago). We used sheets of paper that we had to adjust manually (erasers were invaluable). Some of the figures were adjusted so many times that the paper would wear through.

Years later, on discovering Excel, I reflected how much more effective and efficient completing my accounting class exercises would have been if I had only known of its

existence. Today, Excel is used for far more than accounting. It is used for data storage, data analysis, reporting, graphing, and a bunch of other stuff.

I first discovered R because I was looking for an alternative to Excel for data visualization. What I found was a much more efficient way to analyze and visualize data. R is a statistical programming language. It allows the user to create reproducible analysis — from manipulating data to creating models and graphs.

CSV To Charts

I am going to show a few steps to get you started. The first step is to load data. There are a number of packages for R out there that help to connect you to data sources.

If you find yourself doing any of the following in Excel:

- Loading CSV files
- Lot of index/matching
- Copy & Pasting Data,
- Adding Data Filters
- Creating Charts & Graphs

There are definitely better ways of doing that than in Excel. One is to use R. The learning curve is steep, but in the end, it will be a faster and more flexible way to get some of these things done. Also, since it is code, you can use it repeatedly for other datasets.

This is the code to get a CSV file loaded into R: myFile <- read.csv(file.choose()) I use the file.choose() command to pop-up a file selector. You just choose the data file and the file is loaded. That's it!

If you are using an R IDE like RStudio, then you can view your table by finding it in the upper right-hand corner and clicking on it. You can also highlight the name of the file in the scripting window and click on the Run button (check out the RStudio IDE Cheat Sheet https://www.rstudio.com/wp-content/uploads/2016/01/rstudio-IDE-cheatsheet.pdf). The data will be returned in the Console window.

Tables in R

The table in R is actually called a data.frame and is a set of vectors & factors (vectors generally for numeric data and factors for categorical data). That's important to understand. It can also be helpful for a number of data functions. Generally, when I am exploring data, I just create a copy of the imported data. The reason is, if I make an error, I can just make another copy of the data. To make a copy of a data.frame, you just use the following command: myFile2 <- myFile That's it!

Suppose that you want to sum a column. It's a simple command: sum(myFile2$Sales) That all it takes — the file name, a $, and the column name. If you wanted to get the average of the column, just run the following command: mean(myFile2$Sales) The mean is the statistical reference to the average. Since R is a statistical programming language, it makes sense for that to be the command. If you are looking at the mean, you most certainly will want the standard deviation as well: sd(myFile2$Sales)

Subsetting Data In R

Now what if you wanted a subset of the data that you imported. That can be handled by the subset command: subset(myFile2, Sales > 35, select = c(Product, Sales)) This one is a little more complicated. There are three sections. The first is the data.frame that was created earlier. The second is the subset, meaning the expression of the rows in the data to keep. The third is the columns to return.

Combine Vectors

The columns are enclosed in parentheses after a "c". The "c" is generally used when there is more than one thing in an element of a statement. It basically means to combine values

into a vector or a list, but I like to think of it as “c” for collection, since we are not combining the columns, just returning them.

I’ve added a second column to return called Product, which would most certainly be stored as a factor. If you wanted to use the subset of the data.frame, you can create another data.frame: myFile3 <- subset(myFile2, Sales > 35, select = c(Product, Sales)) myFile3 will be a data.frame with one factor (Product), one vector(Sales) and will only include Sales where the value is greater than 35.

Joining Files

Let’s say that you have loaded another CSV file as a data.frame called allProducts that contains the columns Product and Price and you wanted to add the Sales from myFile3.

Here is one way to do that:

```
Python

# Convert data.frames to data.tables and add a key.

AllProducts2 <- data.table(allProducts, key = "Product")

myFile3 <- data.table(myFile3, key = "Product")


# Join all data from allProducts2 and matching data (by Product) from myFile3
using a left join.

allProducts 3 <- join(allProducts2, myFile3, type = "left")
```

The first thing to see here is the “#”. That is to comment out the text. You want to use this a lot. That is so when you (or someone else) go back to your code, you will know what the code’s purpose is without having to look at the code itself. Second, the code is converted into a data.table, primarily so I can add a key to each table that can be matched int he next command.

Join Command in R

Finally, the “join” command joins all the data from one data.table and the matching data (using the “key” from the prior commands) from the other data.table. If you wanted only data that matched between the two data.tables, replace “left” with “inner”. If you want to return all, use “full”. This is based on the way SQL combines data.

R is really powerful, free (which I love), and, once you get the hang of it, really fun to work with. I hope you check it out, start moving away from Excel, and get to love R as much as I do.

Conclusion

Using a reproducible process goes a long way to ensuring that your analysis is well documented and defensible, as well as saving you time when you need to do it again or augment the current analysis.

P.S.

I’ve been doing a lot more of my work lately in Python. Python is just as good as R, for some things better. It’s always good to keep up on the latest and learn, adapt, and grow.

About the Author

Daran J. Johnson is the co-founder of Fujo, Inc., a digital & performance marketing consulting company. He has been a performance marketing & analytics consultant for over twenty years, working with companies such as Neutrogena, Yes To & Murad. He has a degree in International Business & Economics and is a lifelong learner, always looking to explore new tools, techniques, and ideas. You can get in touch with him by visiting his website at fujoconsulting.com.