DATA_608_Final_Project

Joshua Sturm

May 1, 2018

As can be seen in the source files, I originally used multiple files to get all the information I wanted, before finding a different source that had all of it contained in a single table. I included the years 2010 through 2016.

Due to storage and memory constraints, I had to limit which columns I could use. I tried to choose those which I believed to provide the greatest insight. I did some basic tidying in Excel (such as removing / renaming columns) before loading the file to work with in javascript.

Since my only knowledge of javascript comes from this course, the next part took the bulk of my time. I first tried working directly with my data, before learning that I had to convert each column. I tried for hours to loop through all columns to convert automatically, but when nothing worked, I gave up and did each variable manually.

Once I finally got the file loaded with d3, I ran into several problems with crossfilter. After toiling for *days*, I learned that crossfilter only accepts "long" data, and my file was wide. Through StackOverflow, I learned of something called melt.js, and finally I was able to get crossfilter working - for a single variable. I still haven't figured out how to filter a column in d3/crossfilter - or if it's even possible.

Finally, I resorted to importing my file into R, and melting the entire file... multiple times. My reasoning was that I wanted each variable to have its own column and corresponding "count" column. There is most likely a better way to do this, but I haven't yet found one. The consequence of doing it this way was that each variable was repeated numerous times, and so the crossfilter output was really a multiple of the true values. Using R and Excel, I found out what that multiple was (it was different for each column), and divided by that. The numbers were now close to the real values, although they were in decimal format. When trying to round to an integer, I was getting large errors, which I assume is due to floating point arithmetic error.

The end result is that the dashboard... works. The file is quite large, so it takes ~10 seconds to load, but once it's committed to memory, crossfilter works *really* quickly; the calculations and transitions are near instantaneous. If I had more time, I would further enhance the display with CSS, but I fixed the position of a few graphs to make sure none of them overlap.

I could have made a really nice app in Plotly or Shiny in a fraction of the time (I spent 60+ hours on this project), but I wanted to take on this challenge. Upon completion of this project, and with a working demo, I am glad that I took the more difficult route. I may have missed a few deadlines because of it, but I learned so much from this struggle, that it made it all worthwhile.

Thank you, Professor Ferrari, for an amazing and educational semester!