Assignment 2

Part 1: Gentzkow and Shapiro

The point of chapters 2-8 of Gentzkow and Shapiro's "Code and Data for the Social Sciences" essay/paper is to suggest the best data science practices that one should include in order to not only facilitate your own data analysis projects but also to allow for easier reproducibility. In chapter 2, they suggest that we should automate and write a single script that executes all code. In chapter 3, is to ensure that we store our code and data under version control (meaning that we can easily revert back to previous versions of code in case we accidentally modified and rewritten the original code). In that same vein, we should run the whole directory before checking it back in to ensure that we have not broken anything in our code, either before or after the part that we have edited. In chapter 4, we should separate directories by functions and files into inputs/outputs and make the directories portable. In chapter 5, we should keep our data as original as possible and normalized as a far as we can into our code so that the reader, coder and reproducer understands what the data means.

In chapter 6 and 7, write code that is easy for the reader to understand what is happening without having to reexplain or reproduce more lines of code than is necessary. Furthermore, there is no point in writing documentation if I am not going to change it and keep it up to date. The code itself should be self explanatory and if not, either because the command is obscure or the logic is not evident, then there should be documentation (usually to indicate source material and citations). In chapter 8, create a tak management system that is not email to remove ambiguity.

Automating everything and writing a single script that executes all code solves the problem of replicating my research, both for me and others, and for efficiency reasons. If we were to manually do everything again, there is a chance that we mess something up, use an updated data set, lose some variables when remerging the two datasets etc... which would lead to a different output. Since there is no record of the precise steps that were taken, we cannot answer simple question such as why the number of observations in our table differ form the number of observations in the dataset. Consequently, our numbers will mean absolutely nothing to the reader as they won't understand our process. The second reason is efficiency, if we wanted to rerun a regression using a different specification (using logs instead of levels) we do not want to repeat steps 1-11 of cleaning, merging, recleaning data simply to rewrite one line of code (the regression using logs instead). In addition, its much more efficient for the researchers themselves when dealing with many datasets, logs etc....

Storing code and data under version control, running the whole directory before checking it bac in, separating directories by functions, and separating files into inputs and outputs. Increases quality of life. File version keeps a history of change within a document/research etc.... and it tells you who edited and where. It maintains a single, authoritative version of the directory at all times. However, you need to run the entire directory before we check for changes just in case something got rewritten and breaks your code. Separating by functions and into inputs and outputs makes things cleaner and non redundant (I don't need to redownload the same data set 15 times if I am using it for 15 different projects.

The storing and cleaning of data in a table and keeping it normalized as far into the code pipeline as possible allows two things. The first is that it allows the researcher and the reader to easily understand the data that they are working with. Visuals are the most important thing for a research paper and if I keep modifying a table until it is no longer recognizable it hurts the interpretation of my data. The original dataset has 1000 observations but my table says it has only 100. Why is that? I can't understand why so you cherry picked the data.
Abstract to eliminate redundancy and improve clarity, you want to write as little code as possible as clearly

possible that way the reader understand what I am doing and decrease the possibility of producing an error. The same goes for not writing documentation and making the code readable.

In chapter 8, it's to solve the issue of ambiguity between partners and what tasks need to be completed, by when and if it has been completed.

One problem that might happens is that my partner thinks that they have completed the linear regression that they were supposed to do when they should have done a different regression using an updated version dataset that had accidentally been mauled to death by me but he doesn't know because he can't find the correct log of my data transformation as I wrongly renamed the log in the wrong format and since we can't figure out how to fix it before the deadline we failed to publish the paper. By some miracle, our publisher pushed the due date back allowing us to fix our code. We manage to fix the code and do everything correctly and we send it to our editor to reproduce our code to ensure that our results are correct. He does not understand our code and he runs it in the way that he thinks tit was run by us and gets different result.

I plan to follow these steps as closely as possible and make sure that I complete my checklist.

Part 2: Github

Git is the control system whereas Github is a service that uses the git system. Basically Git is like R the coding language and R studio is the program that uses the R language.

One of the benefits of using git to organize my empirical research is to have file version for when I pulled and pushed files or folders to my repo and being able to pull old versions of it. One common problem is that I forget to send myself the newest r script that I was working on from my laptop to PC, or that I use the old r script when I thought it was the most up to date one.

The most challenging thing about git is the push and pull feature of it. I am able to automatically pull my stuff form the repo after a specified time limts (say 10 mins) but when I commit changes I sometimes forget to push the commit to my repo as there is no limit. I am wondering if there was a way for me to automate it. Simply making sure on my checklists that I have not forgotten to push my commit.

The four git operations are the stage, commit, pull and push operations. Stage: want to add changes to the repo history Commit: yes I am sure that these changes should be part of the repo history Pull: get new changes made on the repo (collaborator or me on different computer) Push: push local change to the github repo.

https://github.com/gaetanoDJ/Titanic