Causal Inference and Research Design
Assignment 2 -Slack and Github Overview

María Alejandra Orjuela Pava

Section 1: Gentzkow and Shapiro

1. Summary of chapters 2-8 in less than one page

Gentzkow and Shapiro (2014) established that one of the main components of performing an empirical social science refers to the fact of writing and executing a proper code which permits to clean and scrape the recollected data, settling up models and statistical analysis. Even though they argued that doing so can be highly self-taught, they did notice that several technical and time-consuming problems may arise as investigations grew bigger. That is why, after getting a set of insights from computer scientists, they decided to write a handbook which, in practicable and understandable terms, showed better ways to work.

The first key aspect Gentzkow and Shapiro highlighted was Automation. They stated that the *interactive mode*, in absence of scripts which showed the main steps taken within data manipulation and quantitative analysis, had multiples drawbacks such as inefficiency and unavailability to replicate results. Therefore, they suggested to automate everything that could be automated, writing a single script that executes all code from beginning to end. Following this advice, they explained that even though there are good reasons for having multiple versions of the same file, the fact of using dates and initials to demarcate them could be troublesome. To cope with it, they introduced the concept of version control; software which permits to track successive versions of a given piece of code. In that sense, they mentioned that, as a repository is set up on a remote server, people could check out and in the willed directory, while the software remembered every version, recording which user did it so. Note that for getting the most of this approach, they enounced to run the entire directory before checking in the changes. The third advice given by them consisted on not only separating directories by function, but also, adopting a subdirectory structure that splatted up files into inputs and outputs, ensuring they were portable and controlled by a single script. In addition to this, both authors argued that it was a must to store cleaned data in tables with unique, non-missing keys. According to them, databases should be self-documenting up to the point where they are able to communicate their logical structure. It ought to be mentioned that keeping data normalized as far as possible into the code is likely to be a great advantage.

Having this in mind, Gentzkow and Shapiro stated that abstraction -turning the specific instances of something into a general-purpose tool- could be useful for eliminating redundancy and improving clarity. However, to ensure that it is done correctly, the practice of *unit testing* might be implemented. Notice that with those objectives in sight, they also recommended to build up a code which is self-documenting; meaning that unnecessary documentation, which is likely not to be maintained, is deleted. According to them, this practice might reduce internal inconsistency, confusion, and unintended behavior. Finally, the authors suggested to manage tasks with a task management system different than E-mails such as Flow, Wrike and JIRA. This can enhance communication and workflow, reducing the problem of asymmetric information within the investigation.

After these advices, they proceeded to incorporate an appendix section entitled *Code Style*. Throughout it they exhibited the most important principles they have learned about writing a good code. This handbook, as a wholesome, illustrated the most important elements of best practice in modern empirical work.

2. **Why do Gentzkow and Shapiro think these elements of modern empirical work are so important? what problems does each element solve?**

Most modern empirical work requires researchers to recollect, scrape, clean and further analyze, through several quantitative and qualitative methods, multiple data sets in order to evaluate hypotheses of diverse nature. Nowadays, there are several programming tools that permit to set up a code which, if executed correctly, may allow the fulfillment of the desired objectives. Despite this, Gentzkow and Shapiro (2014) noticed that within the process, multiple technical problems may arise. Needless to say, those ones are not only time-consuming, but can affect the results derived from the investigation. Therefore, they wrote a handbook which highlighted some key elements to take into consideration when performing this type of activity. They are all better ways to work and are said to be very important as they are likely to: i.) improve the organization and structure of dataset so that it is self-containing, ii.) save up time -spending less wrestling with a messy code, and more on the research problems-, iii.) reduce the costs associated with repeated manual steps du to automation, iv.) guarantee replicability and undo-ability with minimal effort, v.) prevent unintended behavior, and vi.) reduce ambiguity.

When regarding the problems each element solves, it can be affirmed that:

<u>Automation:</u>

The authors recommend automating everything that can be automated, writing alongside a single script that executes all code from beginning to end. Both elements are useful as they counteract the two drawbacks of the 'interactive mode': replicability -no record of the precise steps that were taken to obtain the exhibited results, therefore, absence of authoritative definition of what the numbers in the paper actually are- and inefficiency -constant repetition of all the stages concerned to the building and cleaning process of data-. According to Gentzkow and Shapiro (2014), the fact of learning how to script key steps in data manipulation and statistical analysis, accompanied by automating the largest number of steps, will make the output of the researcher's directory replicable. This will work as a roadmap for any human reader, without being incomplete, ambiguous or out of date.

<u>Version Control:</u>

Gentzkow and Shapiro (2014) stated that there are good reasons to store multiple versions of the same file. However, the fact of naming each of them using dates and name initials is troublesome. On one hand, it may generate confusion and, on the other, it is considered to be poor as far as it enounces barely anything of its content, enhancing the impossibility to replicate results. In order to beat these problems, the authors recommend storing both, code and data, under a version control software. This will permit the researches to track successive versions of a given piece of code, knowing, at the same time, who performed the changes. In addition to this, the software holds the history of changes to the directory. Therefore, if someone wills an old version of it, or a single file, he or she can easily obtain it with just a click.

In the same section, the authors suggest running the entire directory before anyone check in its changes. This prevents the code from experimenting breaks once another person wills to run it, guaranteeing replicability and undo-ability with minimal effort.

<u>Directories:</u>

Within the handbook, it is recommended to separate directories by function, generating -in each of them- subdirectories where files are splatted up into inputs and outputs. This structure solves the problem of re-running the entire data build in case the researcher wants to change a specification. In addition to this, the directories -each controlled by a single script- must be portable. This prevents the code from breaking in case it is run on a different machine, allowing the directory to be worked anywhere with network access.

Keys:

The authors state that it is a must to obtain self-containing databases that communicate their logical structure, keeping them normalized as far as the code permits it. In that sense, having a relational database is the ideal. The fact of following this advice will prevent misunderstandings of the recollected data, deeply knowing the available information.

Abstraction:

Abstraction -action of turning the specific instances of something into a general-purpose took- helps to solve some potential problems linked to the copy-and-paste approach such as incorrectly replacing the input variables within a function. Besides this, due to the generality it implies, it allows the code to be implemented in other investigations which need, for an instance, to perform the exact same process.

This methodology is advantageous as it eliminates redundancy -which reduces the scope for error and increases the value that can be derived from the written code- and makes the code more readable. Note that abstracting without a purpose can lead the researcher to spend a lot of time dealing with cases that will never come up in the work.

Documentation:

It is suggested to document less, only including the helpful comments that provide a roadmap to the reader. If this is done, the possibility that the comments contradict the code -making it unclear which is correct- is minimized. Besides this, it should be known that the problem of internal inconsistency -which emerges as it is tempting to make improvements to the code without making parallel improvements to the comments- is reduced with a self-documenting code. Notice that, according to the authors, documentation can also be used to prevent unintended behavior.

Management:

Finally, Gentzkow and Shapiro (2014) recommend implementing a task management system, stating that E-mail is not an optimal one. If browser-based task-management portals are used, ambiguity within the project's participants will be reduced.

3. **Give an example of the sort of problem that could arise in the course of an empirical project if someone were to fail to adopt these principles.**

In several cases, an empirical project is executed by a relatively large team in which tasks are previously defined and each member must make a great effort so that problems of asymmetric information and moral hazard are avoided. Despite this, it is strongly suggested to follow the elements previously explained by Gentzkow and Shapiro (2014). If, for example, a member of the group does not know how to use a version control software or refuses to implement it, neither checking in nor out the changes he or she made on any file of the directory, several problems may arise. On one hand, that member is likely to be confused over the several versions of files he or she made, and which are named with dates and initials. Therefore, he(she) will probably need to repeat his(her) work, trying to guess and remember the changes that took place. This results in an inefficient and time-consuming inconvenient. On the other hand, if none of the other members of the group observe, within the history of the directory, what he(she) has performed or advanced, work might be overlapped, generating internal inconsistency and confusion. In addition to this, it should be stated that a proper task management system ought to be implemented. If this is not taken into consideration, ambiguity among the team does appear, causing inconveniences in the information's internal workflow.

4. **How do you plan to incorporate these solutions into your own work?**

I must admit that when doing empirical research, my working space alongside with the directories and scripts I implement, are neither the tidiest nor the most organized of all. In many cases, I do create several versions of a file, entitling them unproperly -their name barely says something about their content-. Therefore, I ought to open all of them, trying to 'guess' which is the latest one. Up to this course, I have never heard about Git or any other version controls which permit me to create repositories where I can see the historical changes of my directory. Besides this, even though I do try to have relational datasets, having all the variables labeled and the data being self-explanatory, the documentation immersed in my scripts is way too long.

After this handbook, I am looking forward to learning more about Git, being able to develop certain skills so that I can use it adequately to store all my future work. Furthermore, I will be more conscious of the comments I do include throughout my code. In spite I want it to be clear, I will try to make it self-documented so I can avoid ambiguity. I would also like to learn more about programming so that I am able to minimize the copying and pasting frequency, reducing the scope of error. Lastly, I will be aware of the processes and sections that can be automated so that I do not have to manually repeat all the steps taken to clean my dataset and obtain my statistical analyses.

Section 2: Git and Github

5. **Explain what git and github are used for, how they are similar and how they are different.**

To begin with, it should be mentioned that Git is a distributed version control system which allows individuals to perform several operations to fetch data from the central server or push data into it. This software is installed and maintained on a local system -rather than in the cloud- and it is used for having a historic record of the ongoing programming versions of a certain directory or file. It is installed to manage and keep track of the source code history, being specially designed to work well with text files.

Contrary to this, when examining GitHub, it can be affirmed that it is "an online hosting platform that provides an array of services built on top of the Git System, permitting people to host a central repository in a remote server" (Johari, 2019). This means that GitHub can be considered as a Git repository hosting service -an online database which allows to keep track of and share the Git version control projects outside of the local server-.

Even though they do complement one another, in the sense that Git serves as a software throughout which the researcher can track the changes made within a file, having access to all its versions, GitHub allows it to be portable. Therefore, permitting people to open it in any remote server. This is in fact one of the main aspects in which they differ. Unlike Git, GitHub is exclusively cloud-based, allowing an individual's Git repositories to be remotely accessed by any authorized person, from any server, anywhere around the globe -conditional on having internet connection-. This differential factor, implies that a person can share his(her) code with others, giving them the power to make revisions or edits through various Git branches. Besides this, it should be taken into consideration that GitHub expands upon Git's basic functionality, presenting an extremely intuitive user interface and providing programmers with task-management tools and other additional features that can be implemented through the GitHub Marketplace.

In spite of these differences, it should be stated that both, Git and GitHub, give programmers valuable version-control functionality so that they can develop coding projects without the possibility of messing everything up, losing potentially useful information.

6. **Name a benefit of using Git to organize your empirical research. What types of common problems can occur if you don't use Git?**

Git serves as a version control software, proving the opportunity to not only organize and easily reach all the directories, files, and scripts at a certain repository, but also, to visualize and track all the changes that have been made. As it was mentioned before, there are multiple reasons why someone is likely to create different versions of the same file. For instance, those versions used to be entitled by dates or names, creating confusion and ambiguity of the work performed. Fortunately, this kind of software emerged, bringing along an enormous benefit: automatically recording all the corrections and changes made over time. Git gives each user the power to review and even restore earlier versions, easing the revision and comparison processes if needed. In addition to this, it can be synchronized with GitHub so that all the repositories, uploaded immediately to a cloud-based system, are of easy access regardless of which server is being used or where the person is located. This results quite advantageous when team projects ought to be performed as every collaborator within the repository can have its own branch, making its correspondent changes to the same file. Those branches can later be merged to obtain a common and final file that holds a unique version of the work.

On the other hand, if I do not use Git, I will encounter several problems. Firstly, I will spend a lot of time checking upon all the existent versions of the same file, trying to figure out which one is the latest and which where the changes that I made. It will be extremely difficult, indeed impossible, to go back looking for the code which I suddenly replaced or change. Secondly, I face the chance of losing useful information for my research in case I have not backed up it. Needless to say, those files that are extremely big can't be uploaded on one-drive. Therefore, the use of GitHub through Git can solve this inconvenient.

7. **What about using Git is challenging for you for right now? What step can you take to minimize those challenges such that you can adopt Git for this class?**

To my way of thinking, the main challenge I am currently facing with Git refers to being able to understand how it does work and operate. As I previously mentioned, I have neither heard of this version control nor its advantages, tools, and processes throughout my entire life. Therefore, this has turned out to be a whole new world that I ought to learn as this assignment permitted me to know the benefits, I may derive from it. As far as I am concerned, one way in which I can use Git is through source code at the Git bash. Personally, due to its high complexity, I chose to use the GitHub Desktop as recommended in class so as to minimize this challenge. Nevertheless, I will invest quite a long time trying to understand its vocabulary and structure so that in a near future I can implement the Git bash. In addition to this, I should mention that it took me a while to comprehend the connection, but existing difference, between Git and GitHub. In spite this I noticed that if I take my time, I read the proposed lectures and search on internet for the main insights, I can deeply learn of it. As a consequence, I think those are the actions that I will be performing so that I completely understand all the processes involved in creating repositories, managing files and establishing branches. This is a tool which is going to be useful not only in my academic life, but also professionally and I know that if I look for youtube videos, diagrams and any other kind of visual material, I will be able to make up a clear idea of the processes that the computer and server do and how I can take advantage of those.

8. **Name the four main Git operations. What does each operation do and how does each operation differ from one another?**

There are many operations that can be executed in Git. However, there are four main instructions that are the most important to learn and dominate as they will permit to upload and extract files into GitHub. These are the following:

i.) Stage:

Also known as *add*, permits Git to know which files -that have been previously changed- ought to be uploaded in their current version.

ii.) Commit:

Reassures that the changed files -new versions- which were added are the ones that ought to be uploaded. It permits the user to write a message so as to explain or list the changes made.

iii.) Pull:

Allows to abstract or get out any new changes made on a GitHub repository, either by the researcher's collaborators or by him/her remotely.

iv.) Push:

Permits to throw or get into any committed change that has been locally made in the computer to a GitHub repository.

It should be mention that each operation handles one specific function at a time. However, they are all related to one another as some cannot be executed if the others have not been previously performed. In order to exemplify this affirmation, it should be known that the user must first add rather than make a commit. In case this is not done, Git will display a message highlighting that there is nothing to commit. Furthermore, in order to push a file properly, one must first add, then commit and lastly push.

Lastly, when regarding their differences, it can be noticed that meanwhile *pull* abstracts information from the GitHub repository to the computer remotely, *push* permits to upload any local change to the cloud-based system platform. In that sense, *add* allows Git to know which files were changed, whereas *commit* reassures that those are the ones to be uploaded.

Points 9, 10 and 11 can be visualized by clicking on the following link.

This is my GitHub account which, up to now, contains the requirements highlighted in the syllabus.

https://github.com/MariaAlejandraOrjuelaPava

In addition to this, the link to my own repository named ‘Titanic’, is the following. It contains the main folders discussed in class.

https://github.com/MariaAlejandraOrjuelaPava/Titanic

References

Johari, A. (2019). How to use GitHub-Developers collaboration Using GitHub. Taken from: https://www.edureka.co/blog/how-to-use-github/