**Assignment 2 – Empirical workflow**

1. **Gentzkow and Shapiro**

In chapters 2 – 8 various methods of helping social scientists do empirical work are listed, in terms of managing code and data in an easier and more effective way which has previously been developed by computer scientists and other experts in the subject. The first advice is to use ***automation*** and create a scripts that runs all the code from beginning to end, not only to be able to replicate the results but also because this keeps a record of all the steps that were taken, which improves efficiency and is helpful when adding modifications or running the same project or some of its steps again [1].

The next recommendation is to store the code and the data under ***version control***, as it allows to keep track of all the versions of the code, compare results and revert to previous versions when necessary in a less confusing way, but it must be considered that the entire directory should be run before checking it in [1]. It also records which changes were made and who made them to help solve conflicts. Furthermore, another advice is to use different ***directories***, make them portable and separate them by function. It is also recommendable to create subdirectory structures that separate inputs, outputs, code, and temporary or intermediate files, as this will make debugging easier and will make files accessible when working in multiple projects [1].

Additionally, it will also be helpful to organize de data in relational databases, which is to say structure the database so that variables defined at different levels of aggregation aren’t combined. For this, one should use foreign ***keys*** to normalize the data, so that it will be easier to understand, and mistakes will be less likely to happen. In short, data should ne stored in tables with non-missing keys and the it should be kept normalized for as long as possible [1]. Moreover, ***abstraction*** should be applied to eliminate redundancy and improve clarity. This also helps to reduce the error propagated by copying and pasting, and to use the same function for different projects since it turns them into general-purpose tools, it is particularly useful for tasks that will be repeated often. However, abstraction should only be used when necessary and not taken out of hand [1].

Lastly, regarding ***documentation***, it should be kept up to date in order to avoid confusion and maintain consistency or, to make matters easier, it shouldn’t be written if it won’t be maintained. To help with this, when possible the code itself should be self-documented by changing the names of the variables (for example do that the names declare their functions) and the structure of the code, so that is clear for whoever reads it. However, documentation should be used in some cases such as adding a reference or preventing an unintended mistake [1]. Finally, for ***management*** it is advised to use a task management system, and it should be noted that email is not one of them. This is particularly important when working with other people because a system that allows organized communication and a report of the tasks will prevent conflicts and ambiguity about whose responsibility the task is and what the goals are. Another advantage is that these systems allow to add comments or questions in each task to enable effective communication [1].



Gentzkow and Shapiro believe these elements of modern empirical work are of great importance because they will help avoid several troubles, mistakes and inconveniences when doing empirical research and working with data and code. This will free the researchers time so that he/she can spend more time in their research question and not struggling with the code. For example, ***automation*** solves the problem of replicating results, in which sometimes the exact steps of a project are not stored, and the researcher may not remember what he/she did, so the same results can’t be obtained. In addition, storing the data under ***version control*** solves the problem of changing a script, not liking the result, and not being able to return to the previous version. It also solves the problem of having conflicts in the code when someone else is also editing the code simultaneously.

Similarly, organizing ***directories*** by function and creating subdirectory structures solves the problem of having to create multiple copies of the same directory when trying to change just a fraction of it or when wanting to just a part of it for another project. Moreover, using foreign ***keys*** and normalizing data solves the problem of having databases that are difficult to understand and therefore difficult to use. As for ***abstraction***, it removes the issues of propagating a mistake by coping and pasting different parts of the code and improves the its clarity. Additionally, reducing ***documentation*** eliminates the problem of inconsistency when, for example, some of the parameters used and documented are modified and the documentation wasn’t updated. Finally, using a task ***management*** system avoids problems of ambiguity in responsibility and coordination of tasks when working with other people, and helps keep track of the tasks and their progress so that all of them are completed.



If someone were to fail to adopt these principles, several problems could arise. For example, when finishing a project and trying to replicate years or months later, if the data and the code are not appropriately organized by the guidelines above, it is likely that the person who wants to replicate the study won’t be capable of obtaining the same results because it will be impossible to know which steps where made first. Additionally, if the data is not properly organized or if it has confusing names, probably the reader won’t know the meaning of the variables and won’t be able to use them. Another example of problems that could arise is when you are working in the code simultaneously with someone else, and if the libraries or the tasks are not properly coordinated, the changes made by either of you could cause the code to break if the full script is not taken into account. Finally, if a mistake is made and there isn’t a proper version control of the code, very likely the previous version of it will be impossible to recover.



When working on my own projects, I will try to be very mindful of these solutions in order to make my code easy to understand and manage, I will keep them in mind so that I can apply them from the beginning of my work. In addition, I will read the code several times in order to identify when certain improvements can be added or when modifications in structure and organization need to be made. Also, I will try to show my work to another person and ask whether the steps and the data are completely clear. Additionally, I will use tools for version control such as GitHub in order to keep track of all the versions I make, and I will make a script of the entire code so that all the steps to obtain the results are recorded. Finally, when working with other people I will use a task management system in order to avoid conflicts or misunderstandings.

**Git**



Git is an example of a distributed version control system (DVCS) commonly used for open source and commercial software development [2]. As discussed in the previous section, a version control system is extremely helpful when coding because it tracks the history of changes made in a project. This is useful because several modifications can be made having the confidence that any previous version can be recovered if necessary. Another advantage is that it registers who made the changes and when they were made, so it facilitates communication with teammates when collaborating on a project. Additionally, another advantage of Git, is that it doesn’t need a constant connection to a central repository, and the project can be worked on at any time.

On the other hand, GitHub is a web-based service for version control using Git, it provides the tools to deliver better code through different methods such as command line features, pull requests, code review, among others [2]. It can be used to observe other people’s codes or for other people to comment on your code and help improve it. In short, Git is the version control system, and GitHub is the platform where different people can store their projects and work together with other people using this version control system.



Given that Git is a version control system, it has the advantages mentioned above such as saving time and effort in creating different copies of the code or making redundant tasks when working with others, since it stores all the changes and previous versions of the project. Some of the common problems that can occur when not using Git are that, when comparing different versions of the project, a previous version can be lost if not saved properly. It can also be confusing to keep different copies of the code and sometimes it can be forgotten which changes were made and why or which copy is the updated version.



For me, the main challenge of using GitHub now is that I have never used it before, so I’m not familiar with the commands and the ways in which it works. In order to adopt git for this class I will post all my assignments in my GitHub repository so that I can acquire more experience and be more comfortable while using it. Also, I will use the class slack channels when I have doubts about how to do something. Finally, I will also use internet forums or YouTube tutorials when the need arises.



The four main Git operations are Add, Commit, Push and Pull. Firstly, add is a command that updates the index using the current content found in the working tree and then prepares the content in the staging area for the next commit. In other words, it is the step that goes before commit and is used to add new or modified files to the index. Next, commit is a command that records snapshots of the repository at a given time, these snapshots will never change unless done explicitly, here the recording of all versions of the project are saved. Subsequently, the command Push fetches changes from a remote repository to a local repository and it merges the upstream changes in the local repository, this updates the local repository with all recent changes. Finally, the Push command transfers commits from the local repository to a remote repository, it is the opposite of the pull command. It is used to export and publish local changes in a central repository [3].



Already created it. It can be found in my GitHub account <https://github.com/mariadominguez97>



Already posted it in the class slack channel.



Already cloned it.

# Referencias

|  |  |
| --- | --- |
| [1] | J. M. S. Matthew Gentzkow, Code and Data for the Social Sciences:, 2014. |
| [2] | «GitHub Guides,» 2020. [En línea]. Available: https://guides.github.com/introduction/git-handbook/. |
| [3] | O. S. Zone, «DZone,» 2019. [En línea]. Available: https://dzone.com/articles/git-tutorial-commands-and-operations-in-git. |