**Assignment 2 – Empirical workflow**

**Author: Laura Alejandra Romero Ayala**

**Section 1. Gentzkow and Shapiro - Summarize briefly the point of chapters 2- 8.**

Gentzkow and Shapiro share some insights about coding and managing data into practical terms for empirical social scientists. These are divided in several chapters:

* Automation: allows reproducing the work done at any time (replicability). It is more efficient and can prevent errors. All that can be automated should be automated and written in a single script that executes the whole code- rundirectory.bat recommended.
* Version control: a single store code and data. A rule is to run the whole directory before checking it back in. It is a pain to have many versions of the same and it is also confusing. It is recommended to use a software that allows to ask for a history of changes.
* Directories: all directories should be separated by functions; portable; and files should be separated into inputs and outputs. Sometimes it is useful to have several directories instead of one. Since the data is automated, all will be controlled by a single script.
* Keys: data should be stored clean in tables with unique, non- missing keys and as normalized as possible. It is recommended to work on a relational database, knowing that each table has a key and the variables in a table can include foreign keys. To produce a single array, store raw data in normalized files, construct a set of normalized files (with the transformations) and merge together the tables in the data base.
* Abstraction: turns specific instances of something into general-purpose tools, eliminates redundancy and makes the code more readable and clearer. Warning: abstraction without a reason can lead to wasting time.
* Documentation: the code should be self-documenting and one should not write documentation one will not need. If possible, document less and don’t go over a same aspect twice (it could lead to confusion) and keep the comments up to date. Avoid leading others to error.
* Management: this one is important specially on group projects. It tracks tasks and their responsible with a management system (not email). Asana and Wrike are recommended.

**Why do Genztkow and Shapiro think these elements of modern empirical work are so important? What problems does each element solve?**

They think these elements of modern empirical work are important because they can save time, prevent mistakes, and help those who had never been illuminated with these insights. Also, the elements provide better ways of working and solving problems easily.

* Automation solve the problem that arises when someone tries to re-run a code and doesn´t remember all the process that was made.
* Version control prevents the confusion that arises when one saves many versions of the same document and then there is a confusion in knowing where changes were made.
* Directories permit dividing the code in a way that one does not have to re-run all to make a minor change.
* Keys are a way to facilitate the understanding of data and knowing what every variable is without deeply studying the code.
* With abstraction, space can be saved and errors prevented when running commands with a same general purpose. It can also save time.
* Documentation permits knowing what postulations where used in the code and the literature to be consulted or other relevant information.
* Management allows to avoid confusions on who had to do what.

**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 an empirical work, one would have to answer many questions once the result is published. If one failed to adopt the principles, it is possible that to answer a question or debate a critic, the code is to be run again. If automation wasn´t done, it is possible that when running the code, the results are not the same. It is possible that one does not remember all the steps used in the beginning. It could also happen that a mistake is made or that the revision takes longer than it is supposed to if keys were not correctly used.

In the end, the main problem that could arise is that one ends up all confused in its own code and data set.

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

As logical and obvious as these whole insights might sound, no one had ever taught me these. I plan to download a good software to make version control, since this is the problem that I have faced the more (not knowing what document has what change).

I also plan to use a task manager, since I sometimes have many tasks and end up forgetting the least urgent ones. But mainly, I will try to be more aware on the using of keys, since I use some and a while after I don’t recall the meanings and end up wasting a lot of time.

**Section 2: Github and Git**

**Briefly explain what git and github are used for, how they are similar and how they are different.**

Git is a distributed version control system. This means that you do not have to save many versions of a document because it tracks the changes. Also, Git is optimized for things that economists and data scientist spend time on like codes.

When using Git, every time you save the state of a project, it records how your files look like at that moment and stores a reference to that recording in order to be efficient. It creates a link to all previous ways the project looked like. Thus, Git is like a mini file system.

Meanwhile, Github us different from Git, it is an online hosting platform that provides an array of services built on top of Git system. An easy way of understanding this could be saying that Github is to Git as R Studio is to R: you do not need it to use it, but it facilitates it. Other online hosting platforms include GitLab and Bitbucket.

Thus, Git and Github are complementary to each other, Git allows the control version for projects and Github is where you store projects in repositories and are also allowed to collaborate and network with other people.

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

A benefit of using Git to organize empirical research is that it helps operationalizing the ideas and make them reproductible. Thus, if not using Git, you can end up saving many versions of a same project and getting all confused like Genztkow and Shapiro said in their handbook.

When looking for the final version or for some specific change incorporated in the project, you can get to a point where you do not differentiate among versions and take more than you should finding what you were looking for. Also, Git makes it easier to undo changes if you like more a previous version, meanwhile without the tool, you can easily lose that changes if you don’t save many versions of the project.

**What about using git is challenging for you for right now? What steps can you take to minimize those challenges such that you can adopt git for this class?**

Honestly, for me it is all challenging since I never used a control version system before, so I am just learning now hot it all works.

Some steps that I am thinking to minimize challenges include the use of Github beyond the class. I believe the easiest way to learn how to use a tool is use it as much as possible and learning from all the mini problems you face when doing so.

I will create a repository for every subject of college and maybe even suggest using it at work. This way, I will reduce the chance to mix information and make it easier for myself to search for specific projects in the future. Also, I will share those repositories to the different people that is in each subject, so I don’t mix people either and it can become easier to collaborate in a successful way.

**Name the four main Git operations. What does each operation do and how are is each operation different from one another?**

1. Stage or add is the operation one uses to add changes to repo history. This includes adding, editing, or deleting files.
2. Commit is used to confirm that a change should be a part of the repository history.
3. Pull is used to get changes made on the Github repository. Changes can be made by the owner of the project in another computer or by collaborators.
4. Push is used to include local committed changes into the Github repository.

Operations mentioned above are different because each one has an specific function; you can order them: first you add the files for your project, then when modifying the project you push and commit to those changes and then you can pull that version from another computer or ask your collaborator to pull them so you both work on the same version.

**The first step in your new empirical workflow is the creation of a Github repository (“repo”). You can either do this independently or do this through R functionality. You need to create a github account, then create your first repository called “Titanic”. Initialize with a Readme and create the separate folders that we discussed in class on Monday.**

**Post a link to your repository**

<https://github.com/LauraRomeroA/Titanic>

**Please clone our course github repository on your desktop**

