1.Summary: Code and Data for the Social Sciences: A Practitioner’s Guide

**Chapter 2: Automation**: This chapter main suggestion for practitioners to embrace when programming is to automate everything that can be automated, which means designing programs so that the whole data cleaning process can be replicated automatically from scratch (from the initial datasets). Another option is to create .bat programs that will execute a series of R, Stata, Python, or other files in order as to obtain the same final database every time. This process will ensure that one does not need to repeat the whole process for correcting silly mistakes in programming. It also eliminates the risk of using outdated data frames to calculate results, which would render the results useless.

**Chapter 3: Version Control:** This chaptersuggest two practices. The first is to store code and data under version control. This suggestion aims to eliminate the need of keeping diverse versions of the same database, and the risk of not knowing which database yielded which results. Version control allows programmers to keep only one version of every file, while the software keeps track of the multiple versions of it, so that the programmer may access any previous version of the file if needed. It also helps to keep track of who modified what and why. For version control to work, programmers must commit to automate the file generation process, so that the original databases are never modified and it is thus possible to retrace one’s steps.

**Chapter 4: Directories:** This chapter suggests organizing work elements by function in order to improve the organization of the work environment and be able to find a particular file when needed. In particular, the authors propose to differentiate between input files which should never be modified, and output files, which are a result of the automated processes and are modified by the coding. Finally, this chapter highlights the need of keeping fixed versions of the databases on shared network storage centers, so that results are replicable by anyone with network access.

**Chapter 5: Keys:** This chapter treats the issue of dataset organization. Its main suggestions are to store clean data in tables with unique keys and to keep data normalized as long as possible. Unique keys identify every element within a data frame and help to merge datasets both vertically and horizontally. A normalized dataset contains only observations with unique keys, that are related to variables that are attributes of each observation. Normalized data frames help programmers spot irregularities in the data and allow for easy analysis and merging. One should always store the initial dataset as a normalized dataset, and modify it, only through temporal versions created by the automation process, into inputs for regressions and other procedures.

**Chapter 6: Abstraction:** To abstract is to design general functions that perform a process in the same way everytime. By designing general processes, programmers can save time in repetitive tasks and minimize typing mistakes. It also makes the code more readable. However, it is important to not go overboard with abstraction; it may cause programmers to waste time designing for cases that they will never encounter.

**Chapter 7: Documentation:** In this chapter authors suggest that one should refrain form creating useless documentation, as it may become a confusion source for the code if it is not updated. They propose that code should be self-explanatory, and that fundamental documentation should be always kept updated.

**Chapter 8: Management:** In the final chapter, the authors suggest that group projects should be run in task management systems. These systems allow to keep track of pending tasks, upcoming deadlines, and responsibility allocation. Additionally, task management systems establish dedicated communication channels for each task and thus allow project groups to understand exactly how each part of the project is doing.

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

They believe those principles are important because they substantially improve efficiency to produce empirical social science knowledge. Since most of the work involved in academic production involves writing and debugging code, it is this process where most mistakes can occur. Such mistakes can force highly trained professionals to waste lots of time in reviewing and fixing codes, or even worse, they may completely invalidate the results of an otherwise sound investigation.

Automation and Version control solve the issue of not being able to replicate previous results, allow adjusting the code to correct later found mistakes, and enable researchers to try multiple hypotheses and methods by allowing them to undo changes.

Good directory handling also helps in the replication and improvement of results, as it enables the researchers to understand exactly what files have been created with what purpose, and also allows for external replication of the results of the databases are stored on shared networks. Additionally, having everything neatly organized will help the researchers to write the final document much faster.

Key assignment is fundamental to guarantee that data is being treated appropriately and that the results are actually valid. Designing normalized datasets helps researchers understand what every datapoint means and ensure that their data is consistent (that they are comparing apples with apples).

Abstraction and adequate documentation aim to optimize the researchers’ time and minimize confusion when reviewing the code. They intend to reduce the amount of wasteful code and documentation since it increases the chances of committing petty mistakes and also may confuse external reviewers when validating the process.

Finally, management leverages all the previous elements and tries to optimize the collaboration process among co-authors, by establishing a system for tracking versions, responsibilities, and progress. It helps the project to be developed as smoothly as possible, and minimizes confusing, ambiguous, and non-essential communication that may distort the workflow of the project.

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.

Many examples were given in Gentzkow and Shapiro, however I can give a personal example that happened to my Monetary Policy Making group. We were working on a project to estimate the exchange rate pass through (ERPT) across many countries during the 2008 crisis. Since we did not automate many processes nor established a version control system, we had a dropbox filled with endless versions of the same dataset and with many copies of slightly different graphs and tables. We did not even name them on a date basis, but rather just used the word new, final, definitive final, etc, to differentiate among them. This of course made the whole process much more cumbersome and lengthier. Nevertheless, we still managed to finish the project and obtained remarkable results.

The biggest problem came when we presented the project to the professor. He was very impressed by the results but suspected we had committed several small mistakes in the calculations of the impulse-response graphs. We explained to him that we had indeed committed the mistake he suspected but had corrected it for the final version of the project. However, under further revision, we had used the wrong graphs instead of the right ones, and we could not even find the right ones in the dropbox. He deducted a few points from our grade and said it was alright, but we could not help thinking that we would have gotten a perfect score if we had been a bit more organized.

I have tried to adopt some of the principles outlines by Gentzkow and Shapiro ever since.

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

First, I would like to state that I already implement many of the aforementioned principles. In particular, I try to automate as many processes as possible within the code such as the data cleaning process. I am also very careful not to modify the original datasets but instead use automated programs to generate new temporary versions to work with. Likewise, I like to create general function for processes that will be undertaken several times, and I try to make my code as self-explanatory as possible. Finally, as an economist, I am used to using unique keys to identify observations and have experience in database handling.

I still need to incorporate the principles on directories, version control and management. I intend to implement the directories solutions right away, as you will be able to see on my github repository. However, I anticipate that the main problem will be sticking to it during a real project (Such as the 3rd assignment). For the management principles, I will try to learn how to use slack during this course. However, as Gentzkow and Shapiro admitted, management problems are really small when working alone. What I will try to do is to keep an organized log where I update the progress on various tasks within each project, so that I will get used to do it before I try to implement it in a real group project. Finally, I will start incorporating version control into my work beginning in the 3rd assignment, where I intend to follow closely the guidelines of Genztkow and Shapiro.

5.Create a new section in the document you used to answer questions 1-4. Briefly explain what Git and Github are used for, how they are similar and how they are different.

Git is a software that manages version control. It is fully installed locally on a computer and allows the user to create repositories for local projects and to keep different versions of the same file within the file. It allows the user to create branches in which he/she may modify files, while keeping the master version of the file unchanged. When the user is satisfied with the branched file, he/she may merge it into the main branch, while keeping earlier versions of the file. If the user is dissatisfied with the result, he/she can erase the branch and keep working with the initial version of the file.

Github on the other hand, is a hosting service for remote repositories. That means, Github allows users to store their files in the Cloud and thus facilitates cooperation between teammates. It also offers some tools for project management and allows user to participate in collaborative projects. Github allows users to access an updated version of the online repository at every time. It is important to highlight that GitHub enhances Git functionalities; it works using the Git language. It has the same version control functionalities but allows for different authors and remotely hosted repositories.

Summarizing, Git is mostly used for local version control, which is best for managing private or confidential data. Github is used for collaborative projects in which coordination is fundamental.

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?

One of the benefits of using git to organize my empirical research, is the possibility of being able to keep track of multiple versions of the same file without having to create many files. My computer is filled with many versions of similar files, so I think using Git will accelerate my work rate or at least facilitate finding the files I need.

Many kinds of problem may occur when not using Git for version control. The most important problem may be erasing or modifying a file that was used to obtain the desired results. This will render them irreplaceable and thus force you to start coding form scratch in order to correct an error or rerun an estimation. Another problem could be not being able to find the right file to match it with the right code, and thus having to work from scratch every time you lose track of your current version. Not using version control also makes projects vulnerable for overlapping modifications from different team members which may bias the results or even render files unusable.

7.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?

Almost everything is challenging because I am fairly new to version control software. The most challenging part may be managing the branches interface, since I am still unsure if I can choose to keep ‘parts’ of a branch that I liked and discard the rest. Aside from reading the of Genztkow and Shapiro book and the Pinter slides (which I already did), I think the only thing I can do to better understand how Git works is using it. I will try to explore all functionalities in the 3rd assignment and ask Professor Scott for help in the office hours when I get stuck at something. In the end, I believe experimenting with the program will show me its advantages and convince me to adopt it not only for this class but for all of them.

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

Commit: This operation creates a version of the file you are working on, within the branch you are currently working on. It is like taking a picture of the file, and you will be able to access that version of the file should you need to.

Merge: This command unifies the selected branch and the main branch of the project. In practice, this means to save the changes made to the branch file and turning them into the mainstream version of the file.

Pull: This command fetches remote changes on the current branch into the local clone and merges them into the current working files. It updates your personal version of the repository with the changes made by others.

Push: Uploads changes from all local branches to the respective remote repositories, thus making them available for all users with access to the repositories.

9.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.

Done!

10.Post a link to your repository

https://github.com/Sebastian-Benavides1999/Causal-inference-course.git

11.Please clone our course github repository on your desktop

Done!