Archive

Analysis Scripts

Have you ever run data analyses interactively in a program by entering directives on a command line and cutting and pasting the results into a document without saving the list of commands used to produce the output? Most of us learn quickly that this is unacceptable for a variety of reasons and Im sure you can list at least two if you think about it for a short moment. Thus, we learn to create and save a script that documents the set of commands we executed to create our results. Every analysis involves editing an existing script or starting a new one. You might save a revised script under a new name to preserve a record of how you achieved earlier results. Maybe append a date to the end of the name of the script to identify different versions. Soon your directory has XX files and whatever naming convention you use you will still find that it is difficult to identify what script produced a specific set of results and/or what set of results is the definitive set.

A **version control system** like GitHub might help you to keep your own files orderly. It can also facilitate collaboration, provide a way to keep your files secure, and make sharing code easier both pre- and post-publication. The tool doesnt solve all your problems by itself; you also need a plan. But before you can develop a plan, you need a conceptual familiarity with version control.

[Need to add a conceptual description of version control system. Would not cry if I could find something already available online.]

Unfortunately, even with a version control system in place, you still need to do some **planning work** to stay organized. The key problem remains: how do I know which files produce which research results and which result results are the ones in the most recent version of the paper?

Here is where your code structure can help. We suggest that you have a main script that executes the data creation, analysis, and production of publication-ready tables and graphs with a push of a button. The main script executes a script that creates the data and another script that conducts the analysis. The script that conducts the analysis identifies the specific scripts needed for the analysis. This is nice in theory and in practice it is a flawed system. We rarely know when we have the final version of an analysis. We usually want to keep various versions of our analysis in case we want to go back to a previous version after testing an alternative and finding it unsatisfactory. My projects are developing systems for managing themes and variations using git branches and we will report back once we have something shareable.

Manuscripts

Sometimes (who am I kidding?) we have a few false starts when writing the introduction or background sections of a paper. Occasionally out comes a brilliant paragraph that would help provide strong conceptual foundations for a different paper than the one we are currently trying to write. Learning to write is, among other things, learning to delete text that doesnt belong. Having had the experience of deleting text later fondly remembered as the best thing I ever wrote, I am reluctant to completely erase it. I might instead move it to the bottom of the document and it will get carried forward to new versions. Alternatively you might save it in a separate file. If you save it under a different file name, you'll want to keep a document that indexes your saved files so that you can find the deleted text later when you want it.

If you use acloud storage systemlike Box (UTBox) or Dropbox to store your files, the system likely automatically keeps past versions of your files so that you do not need to save files under a new name to keep an archive. For example, UTBox saves up to 100 previous versions of a file. You can see the version history by going to your folder inUTBoxand clicking on the file. A version history will appear on the right side of the page. If you want to keep a version more than 100 saves ago, you'll want to archive it somewhere using a different name, but this would likely reduce the number of files on your harddrive. This approach does not make it especially easy to see what changed between versions.

Allowing computers to keep a version history helps to keep directories clean and reduces the risk of having to reconcile independent revisions of the manuscript. In the past, I and my collaborators usually created new versions of a manuscript with every editing session, often in combination with changes tracked. This belt and suspenders approach helped to ensure that we don't lose text, but it has some downsides too. First, track changes makes the file messy and difficult to read. It's, of course, possible to look at the document without changes tracked by selecting "no markup" on the "review" toolbar, but it is not possible to show just the changes since the last time I edited the file unless everyone accepts all changes before starting their writing session. In my experience, co-authors are slow to accept changes and therefore it is still difficult to tell what changed between this draft and the previous one. Second, my directory became crowded with multiple versions of files. This increases the risk that I will edit the wrong version of the file. For example, maybe yesterday a I looked at a version of the text as it was last month, mindlessly corrected a typo, and "saved" it. Then my co-author starts her day by sorting the files by date modified and selects the most recently edited file to start her writing session. Or maybe I've anticipated this problem and have named my files in a way that allows me to find the most recent file by sorting by name and not just by date of most recent save. Assuming that my collaborators and I execute this system perfectly, I still have a lot of files with similar names sitting in my cluttered directory and it takes effort to make sure I open the right one. For this reason, I prefer to use a cloud storage system and let Box or Dropbox automatically keep a version history.

Alternatively, you could keep all of your text files, including the text of the article, on GitHub. You check in your text like you check in your code and you can use git diff or your GUI tool to see changes between versions of the text. You have just one version of the file, the most recent one, in your directory. Thus far, for me, a barrier to keeping text files in GitHub is that I cannot link references in a text file to the zotero database. There are some potential advantages of keeping text in GitHub however. For example, your archive would allow you to see the contemporaneous state of the code and state of the manuscript and this would indirectly serve as some documentation for the code. If we overcome the problem with references and develop systems for keeping our text in GitHub, we will report back.