File Structure

Your computer has a directory structure with a base directory and sub-directories. You might have a sub-directory for each project you are working on and, within that directory a sub-directory for each paper within that project. For your research you will need to store many different kinds of files including documentation and notes, original data, created data, scripts for creating variables, scripts for data analysis, log files, and results files. The specific organization might be different for each project and even for each collaborator on a project. Nonetheless, you should have a plan for organizing and documenting the location of each element.

One approach to organizing your project might be to create a sub-directory for each kind of document. Further, naming directories something usefully descriptive can make it easier for you to find the files that you want. The specific organization will likely be shaped by how you are using cloud storage (see below). The set of directories that you will likely need for each paper/project include:

> Original Data – ideally everyone on the project has access to the exact same data. The best way to make sure this is true is to be working from the same copy of the data and this means that you will use cloud storage or a network drive to to house it. Definitely write-protect this directory. It is also a good idea to include a document in this directory describing where the data came from and when they were downloaded. An alternative might be to keep the original data in your GitHub repository. I don't do that because the data I use often includes large data files and I don't want a bloated repository.
>
> Manuscript files – You'll want a place where you store drafts of the manuscript. This is also where I store the Project Master Document described in Documentation. This directory is also in a space that collaborators can reach. Some time in the future I might start to store the drafts of the manuscript and other notes in by GitHub Repository and that will mean that I would NOT store these files in a directory that can be accessed by collaborators. Instead, collaborators would pull these files from the repository, but that works only if my manuscripts and notes are text files and not stored as word documents.
>
> Scripts for processing and analyzing data – These are archived using GitHub. Note that each project has a setup script that calls a personal setup script for each collaborator. The personal setup script sets macros for the location of each of the directories on the project. There are some examples of this in the tech tips (but it needs some development).
>
> Created Data – these are data files that I will use in the analysis. Often they take a long time to create and I don't want to have to rerun all of the code that creates the data every time I do an analysis. So, I have a directory for created data files. I do not share this directory with any project members. Each one of us has our own copy of the created data that we create using the scripts that are housed in our repository. This makes it so that each project member is replicating the results independently. To make sure we stay in sync, I will typically pull all changes from the repository when I start work for the day and rerun the code to create the analysis data files. Then I might comment out the portion of my main do file that runs the code to create the data for the rest of the day to save time.
>
> Results – this is where the automated analysis scripts put the results, including output from markdown files. I do not share this directory with collaborators so that each person on the project replicates the results independently. This practice aligns with suggestions in Christensen, Freese, and Migel's Transparent and Reproducible Social Science Research to not save statistical output but instead save the data and code that produces that output. Of course, our current systems require that you have at least one copy of your results to submit for peer review.
>
> Log files – I put this in a directory separate from results just to keep my results directory relatively clean (but it is still messy). Log files might have dates in their names, but that will mean storing a lot of log files. This directory is not shared and it's not in the repository.

When working solo on a project, most will have all of their files for a project within a subdirectory with the project's name. For collaborative projects, as well as projects for which you want to make your code available to enable others to replicate your work, I recommend that you have separate directories for your scripts, original data, analytical products (results, created data, and log files).

For collaborations, keep your original data in a location that is accessible to everyone on the project. It is true that each person could create a copy of the original data to put in their own personal research space. When the project data are updated (e.g. you add a new variable to the IPUMS extract, a new batch of responses to your survey comes in, or the federal agency that produced the data provides a new version that corrects an error in an earlier release), then each project member will need to remember to copy the data into their space every update. Otherwise, different members of the project will be working on different versions of the data, a potential source of confusion. To avoid this, your code could access the data in a shared project directory. When the data file is updated, the project manager should archive the old version of the data in a way that preserves information about its origins, contents, and version. The new working data file should have the same name as the old file so that it can be seamlessly accessed by already established scripts.

I also recommend that you keep your scripts in a directory that is linked to a code repository that is backed up to the cloud, like github. I discuss the benefits of using a code versioning system elsewhere. Having your scripts in a repository has important implications related to file structure. First, you will want to avoid putting your data in the code repository, unless your data file is very small and will not change. To remove the temptation, you could either add the data to the git ignore list or keep your data separate from the git repository. Second, keep the products of your code out of the repository as well. One reason why you want to keep the original data and analytical products out of the repository has to do with how versioning systems store your changes. In short, git compares the contents of the old version with the new and stores the difference. If the files are binary (e.g. a stata data file, excel file, word document, image) git is unable to usefully make the comparison and will store the whole file. Overtime this will make your repository unwieldy. Another reason is that the data products can get out of sync with the code if some members of the research team are committing only the scripts, which I' m arguing is the better practice. Trust the process. If you need to go back to an earlier research product, you can checkout the version of the repository that produced that product and re-run the code to reproduce it. If your code cannot produce the data product, then the product isn't documented and the system isn't working as designed.

Use subdirectories within your script directory to organize your code. (for example, this project has subdirectories for the R. and stata versions of the file organization scripts). When your moving files across directories, use git mv rather than your operating system to move the files. In this way the history of the file is preserved and git wont reinstall the old version of the file in its old place when you sync-up your repository to the cloud.

An example of a setup for reproducible results is here: https://github.com/kraley/workflow (current ly only for stata, but I'm working on a version for R).

Naming files

Some people advocate for including a descriptive prefix for all files related to a paper or project. For example, I might have a project investigating trends in divorce and so all files related to that paper should start with "div." I once used this approach but over time I've decided that it is a waste of characters in the name of the file. The directory name tells me what project the file is for. Instead, I give files a name that describes the role of the file on the project, like PAA_ExtAbstract. This extends to the scripts that analyze the data. These are also stored in a project (or paper) specific directory and are named according to function rather than the paper topic. If you want to share a file you can point someone to your GitHub repository for the project and so they have the file in context.

You might develop conventions about what to call each piece of the documentation for your project. Such as

Project Master Document - "master.docx" or "master.txt".

Notebook - "RunningNotes.txt"

Code Master Document – README.txt of README.md. GitHub will kindly create one of these when you initiate the repository.

For a current project we are naming our analysis files for the part of the paper that they produce. Doing that hasn't worked great because we have many files named Table 2XXX and only one actually produces the real Table 2. It's true that the XXX part of the name gives us some clues as to which file is likely to produce the actual Table 2, but it has been a source of mild confusion on the project. So, I don't (yet) recommend this approach.

Cloud Storage

On another page I discuss the different kinds of cloud storage, but this relates to your file structure as well. With the File Structure described above, I keep the Original Data and manuscript files in Box or on a network drive that my collaborators can access. I keep scripts for processing and analyzing data in the GitHub repository. Created Data, Results, and Log Files are all stored in my own personal directories and not shared with my collaborators who are expected to independently replicate results.