Contents

What is reproducibility?

What is reproducibility?

"reproducibility refers to the ability of a researcher to duplicate the results of a prior study using the same materials as were used by the original investigator. That is, a second researcher might use the same raw data to build the same analysis files and implement the same statistical analysis in an attempt to yield the same results. . . . Reproducibility is a minimum necessary condition for a finding to be believable and informative."

NSF 2015, cited in Goodman, Fanelli, and Ioannidis 2016, Science Translational Medicine Vol. 8, Issue 341, pp. 341ps12

What is reproducibility?

- the ability to arrive at the same results obtained in an investigation **given the same data**
- the person reproducing the results can be the original researcher in the future, or some other researcher
- different from replicability: ability to duplicate the results of a prior study following the same procedures but gathering new data
- requires discipline in organizing and distributing research materials

What is reproducibility?

Why strive for reproducibility?

- it is a minimum condition for results to be understandable and reliable
- facilitates work with coauthors
  - coauthorship is increasingly common
- increases efficiency and scalability of one's own work
  - research projects can be complex and painstaking work
  - given long stretches of time to publication, one is likely to have to revise work when one does not remember all steps carried out so far

What is reproducibility?

- we'll see two strategies to increase reproducibility of our research
  - automatization and directory organization
  - version control (with Github Desktop)

Directory organization

Directory organization

Suppose we just gathered data from an experiment. Instinct might move us to proceed to analysis as follows

- open program to analyse data
- import data
- run regressions
- store results to then write the report

Directory organization

That is the antithesis of reproducibility

- no record left of how results were obtained
- a revision in the future requires us to repeat all steps

Directory organization

One solution is to write scripts

- programs for the automatic execution of tasks
- we should use them for as many tasks as possible
  - load data
  - clean data
  - analyse data

Directory organization

Example of script

Directory organization

The goal is to use scripts to automatize the process starting from loading the source data all the way to generating the results

Being scripted, the process is reproducible.

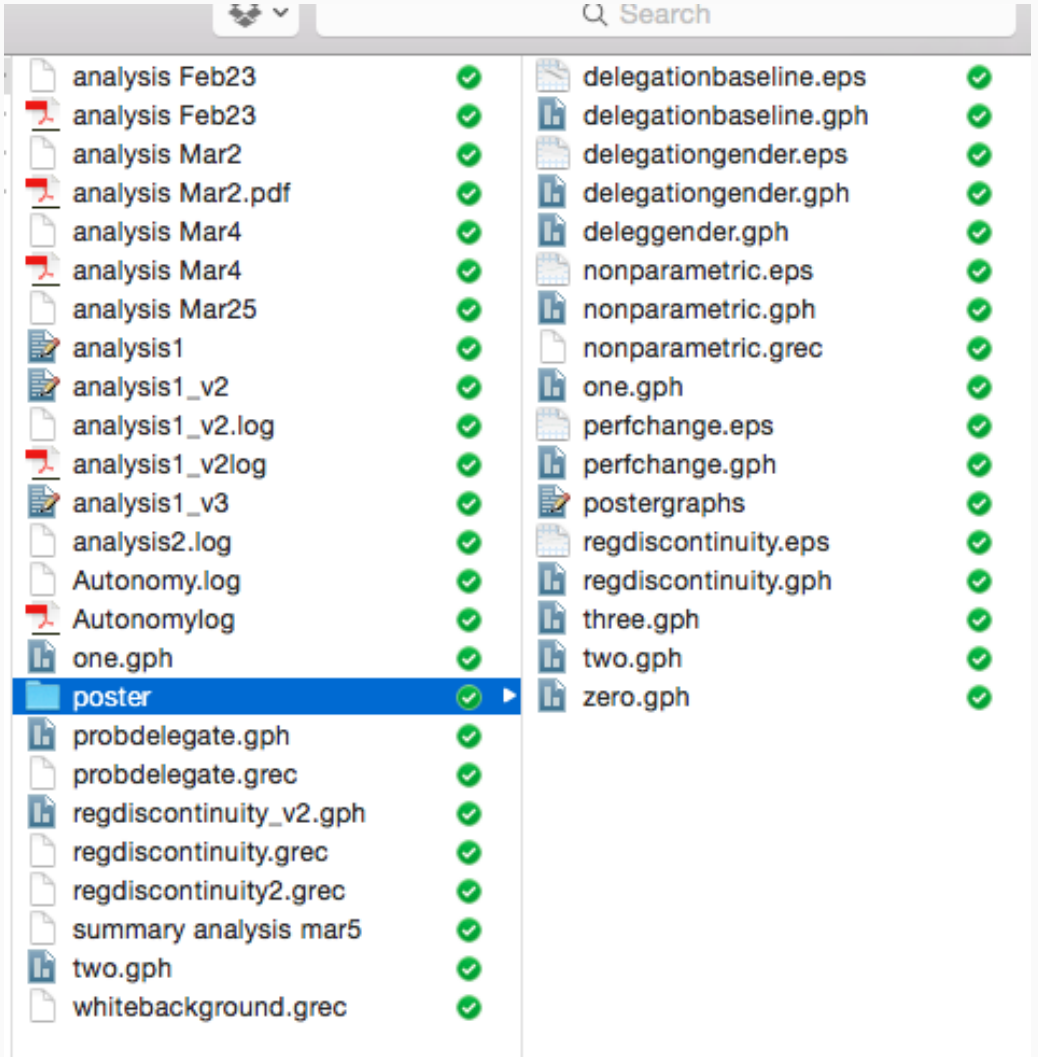
Directory organization

Such system requires organization to avoid confusion over

- which file depends on which
- if new data is obtained, do the scripts run correctly?
- if a script replaces the data, are the original data preserved?

Directory organization

Chaos



Directory organization

Unusable code after so much time invested

```
. do "/var/folders/n8/ydk3gnf112ngh7f11gxn_m900000gn/T//SD01180.000000"

. use "./dataset.dta"
file ./dataset.dta not found
r(601);

end of do-file

r(601);

.
```

Directory organization

Tips to manage research workflow and directory structure:

- Scott Long: *The Workflow of Data Analysis Using Stata*
- Gentzkow and Shapiro: A Practitioner's Guide *video*, *texto*
- *Software Carpentry guide*

Directory organization

Tip #1: Treat data as "read only"

- do not edit manually (eg. excel, Stata editor)
- do not rewrite
- this ensures data integrity and allows one to see how the source dataset is modified (via scripts) to produce another dataset

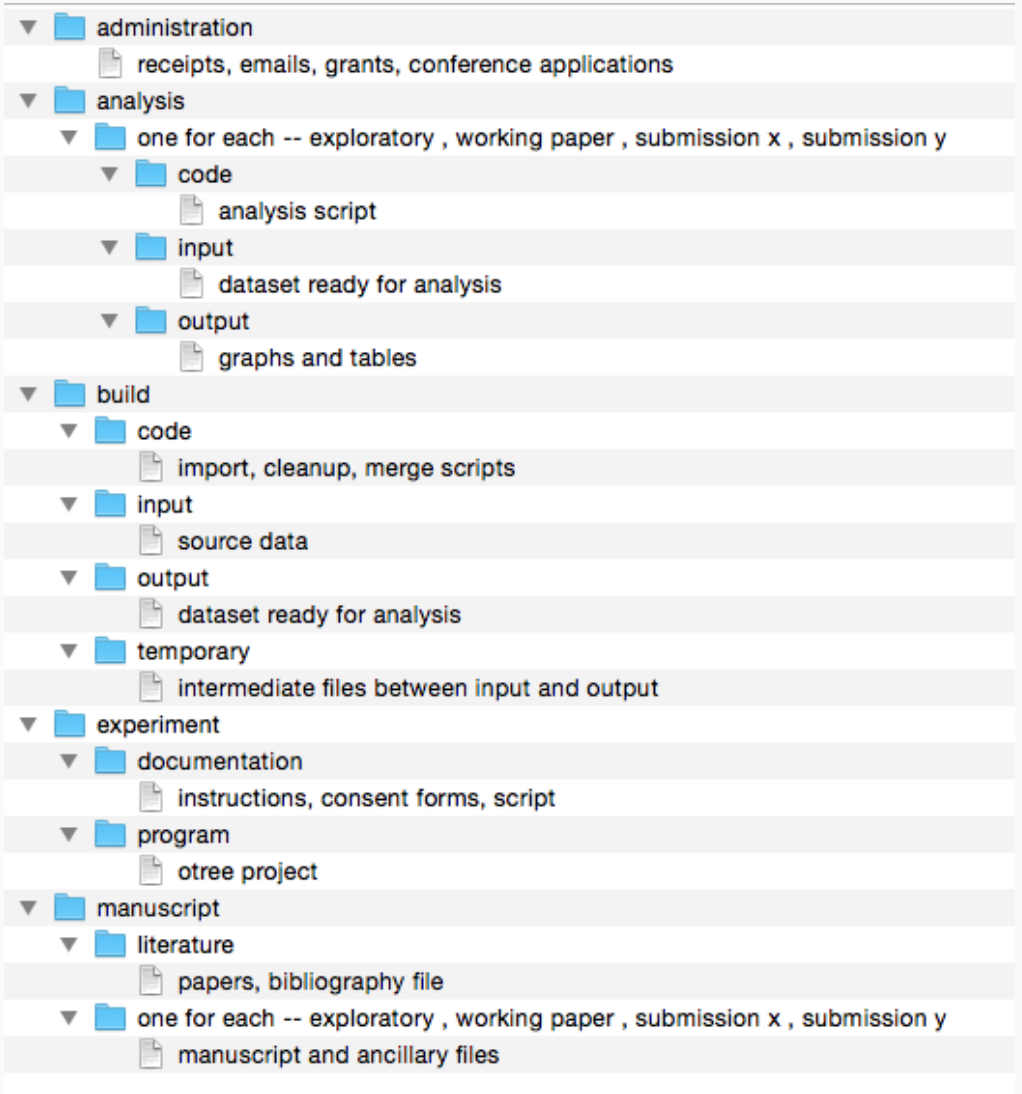
Directory organization

Tip #2: Create different subdirectories for cleaning and for analysing the data

- scripts for cleaning take the source data, clean it (add/remove variables, combine datasets), produce a new dataset for analysis, and store it as a separate it file
- scripts for analysis take the data ready for analysis and perform the analysis
  - can edit the data but never store data
  - produce and save graphs and tables

This way the entire process from source to results is reproducible, and the analysis can be re-run without have to redo the cleaning

Directory organization: one suggestion



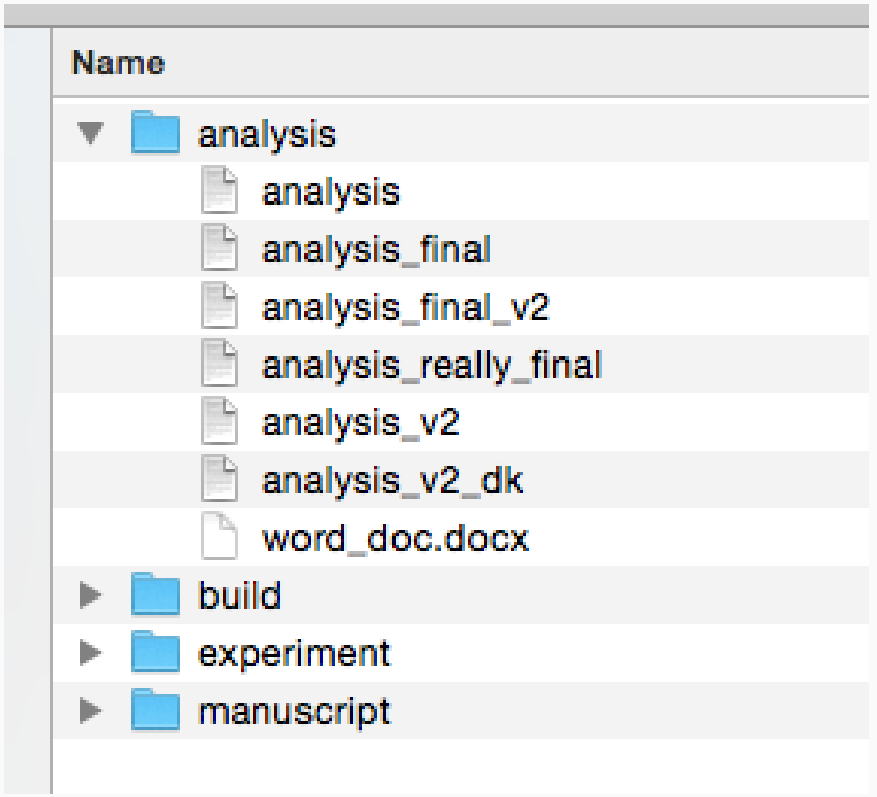
Version control with Git using Github Desktop

Version control with Git using Github Desktop

- automatization and directory organization are steps toward greater reproducibility
  - remove manual interventions on the project
  - leave record of all actions
  - compartmentalize the project, make clear dependencies between files, allows publishing/sharing of only certain parts of the project
- even with automatization and good directory organization, chaos inside any directory can render the whole process useless

Version control with Git using Github Desktop

Which is the final analysis, and how is it different from the others???



Version control with Git using Github Desktop

Answer: Version control

- a tool to take a "snapshot" of the project at any moment
- keeps track of changes from one snapshot and another, as well as who made the snapshots and when
- allows recovering of project to its state at any snapshot
- many version control systems exists
    - we'll use Git

Version control with Git using Github Desktop

Git vs. Github vs. Github Desktop

- Git
  - a system to do version control
  - the most complete and flexible way to do Git is through the Shell
- Github
  - a company that hosts Git repositories
- Github Desktop
  - an app created by Github do perform basic Git functions in a graphic interface, without using the Shell

Version control with Git using Github Desktop

Brief intro to version controlling a research project with Git using Github Desktop

1. sign up to a Github account
2. install Github Desktop
3. track a directory
4. upload directory to Github
5. retrieve previous versions of a project hosted in Github

Version control with Git using Github Desktop

1. Sign up for an account on *Github*
   - Github offers private repositories for a fee, but students and researchers can *request them for free*

Version control with Git using Github Desktop

2. Download *Github Desktop* and follow the installation instructions

- once installed, open the app
- go to Preferences/Accounts to link the app to your account
- enter username and password in Preferences/Git
    - enter the email provided by Github to avoid making your personal email public – for this go sign in to your *Github account*, click on the icon on the top left, and then Settings/Emails

Version control with Git using Github Desktop

- create folder "*my_project*" on the Desktop
- create a simple *.txt* file and save it as *analysis.txt* inside *my_project*
- in Github Desktop, click on *Create New Repository*
- Name: *my_project*
- Local Path: Desktop
- *Create Repository*
- a new text file named *analysis.txt* is now being tracked, as shown in the History
    - in fact anything done in the *my_project* directory is being tracked

Version control with Git using Github Desktop

Changes to the project do not get saved as a permanent snapshot until we do a "commit". Creating the repository was our first commit. To do another one:

- make changes to the analysis.txt file
- at the *Summary* bar over the *Changes* section, enter a brief but informative description of the changes to the project
    - ejg: added a second line to analysis.txt
- *Commit to master*
- Now *History* shows two snapshots (commits)

Version control with Git using Github Desktop

We can make multiple changes to the project before taking a new snapshot

- more changes (add and remove lines) to analysis.txt
- create subdirectory *manuscript*
- save a word document into this subdirectory
    - binary files such as .doc files are tracked similarly, but Git cannot explicitly show us the differences between versions
- do a new commit

Version control with Git using Github Desktop

- so far our project, with all its versions, is stored only locally in the computer
- it is advisable to also store it in a remote location
  - as backup
  - to share the project with coauthors and the general public
  - to retrieve previous versions of the project without using the Shell

Version control with Git using Github Desktop

To move ("push") the repository to Github from Github Desktop:

- *Publish Repository*
- *Name*: my_project
- Now all our project snapshots are hosted in Github
  - the history of the versions is in the *Commits* section
  - we can download any version of the project
  - others can get a copy ("fork") of our project

Version control with Git using Github Desktop

Finally, to download ("fork") someone else's Github repository:

- we'll download my *suggestion for directory structure*
- To download it from the website:
  - Clone or download / Open in Desktop
- To download it from Github Desktop:
  - *Clone a Repository*
  - URL: *https://github.com/davs-econ/research_project_structure*