# Version Control

## Data Science Tools Workshop

Fabien Forge

08/10/2021

## Who I am

- PhD in Economics from the University of Ottawa
- Currently a Postdoc at uOttawa and Part-time professor
    - I teach econometrics (McGill) and data science (uOttawa)
- Environmental economist
    - applied work
    - machine learning

## The tools I am using

- My first language was Stata
  - Stata is terrific language and extremely capable
  - It is limited by its proprietary nature
- When I started the research part of my PhD I dived into Python and R
  - I needed Python to handle weather data
  - I also discovered the many capabilities of R
- Today my preferred language is:
  - Python for machine learning
  - R for presentation, website, Github
  - Stata and R for causal inference regressions

## Motivation

- Data science is a portmanteau word which covers many disciplines
- The tools used in data science are generally derived from the needs of developers and people using data in production
- This means that not all tools will correspond to our needs
  - We rarely receive new data everyday that we need to ingest
- But some of these tools can be very useful to us

## Motivation

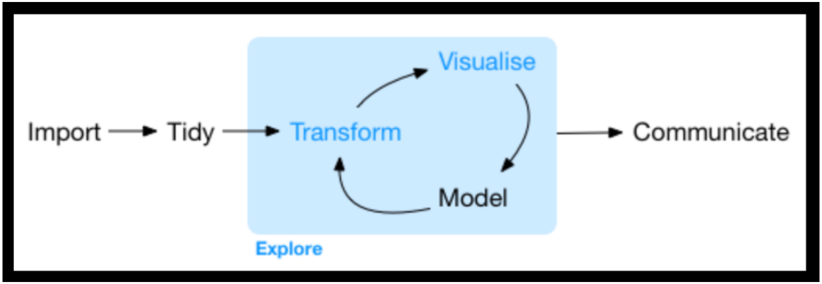


Figure 1: Source: r-bloggers

## Open source

- Unlike Stata, R, Python, Google Colab, Github etc. are open source (or at least free)
  - Although for Github when services are free you may be the product
- What makes open source so great is that it offers:
  - great modularity
  - complementarities
  - community support

## (Open) Sources

- For today I was inspired by:
  - Grant McDermott from University of Oregon
  - Free Code Camp Git and GitHub for Beginners
  - Hadley Wickham
- All these resources are available for free
- I'm happy to point you towards more resources if you want

## The tools - Version control

- We may not use new data every day but we are revisiting our scripts often
- sometimes months pass before we go back to them
- sometimes researchers come in and out of projects

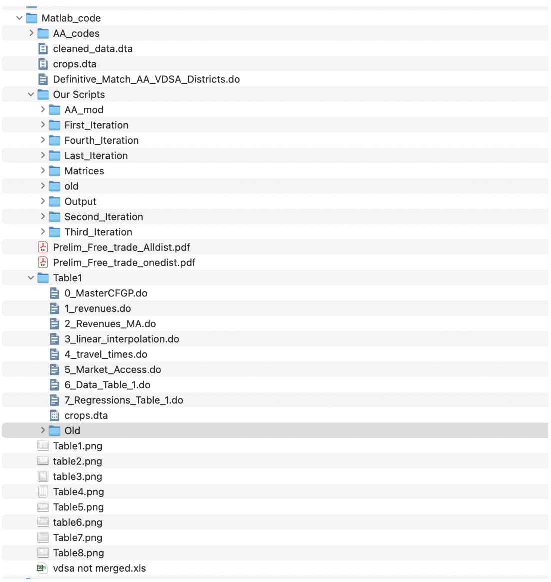
## The tools - Dynamic Documents

- Code does not have to be detached from text, equations, or the tables and figures it produces
- Being able to bring everything in a single document can be very efficient
- Jupyter and R-markdown works both with Python, R or even Stata
- You can do many things with them including this presentation or personal websites!

## The tools - Unit testing

- Code always does exactly what it is told to do
- It's not always the same thing as what want it to do
- The need to publish data along with papers makes the need for robust code even greater

## Looks familiar?



```
Matlab_code
  AA_codes
  cleaned_data.dta
  crops.dta
  Definitive_Match_AA_VDSA_Districts.do
  Our Scripts
    AA_mod
    First_Iteration
    Fourth_Iteration
    Last_Iteration
    Matrices
    old
    Output
    Second_Iteration
    Third_Iteration
  Prelim_Free_trade_Alldist.pdf
  Prelim_Free_trade_onedist.pdf
  Table1
    0_MasterCFGP.do
    1_revenues.do
    2_Revenues_MA.do
    3_linear_interpolation.do
    4_travel_times.do
    5_Market_Access.do
    6_Data_Table_1.do
    7_Regressions_Table_1.do
    crops.dta
    Old
  Table1.png
  table2.png
  table3.png
  Table4.png
  Table5.png
  table6.png
  Table7.png
  Table8.png
  vdsa not merged.xls
```

## Git to the rescue

- Git is one of the tools one can use to do version control
- It is a way to store information and keep track of modifications in your code
- Paying the fixed cost of learning this tool can prove a very good investment

## GitHub

- You can think of Github as being built on top of Git
- There are other competitors (Bitbucket, Gitlab. . . )
- One can absolutely use Git without having access to Github but the latter offers nice additional features

## Github for economists

### From software development...

Git and GitHub play a major role in software development

### ... to scientific research

- Of course version control helps for organizing your code and work colaboratively
- It is also a key component of open science and reproducibility
- Journals have increasingly strict requirements regarding reproducibility and data access.

## Github

### What is Github?

https://github.com/

## Github desktop

### What is Github desktop?

https://docs.github.com/en/desktop/contributing-and-collaborating-using-github-desktop

## Github lingo - Repository

- A **repository** is usually used to organize a single project.
- Repositories can contain folders and files, images, videos, spreadsheets, and data sets – anything your project needs.
- Usually it includes a README and you can provide a license file.

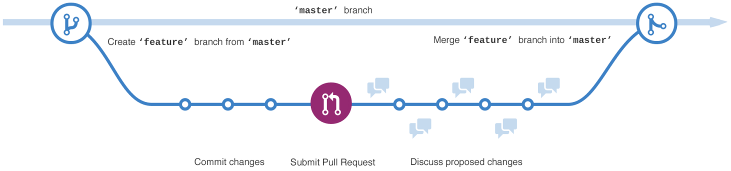
## How to create a new repository

- In the upper right corner, next to your avatar or identicon, click and then select New repository.
- Give a name to your new repository
- Write a short description
- Select Initialize this repository with a README

## Github lingo - Main and branches

- By default your repository has one branch named **main**
- You should think of main as your best current version
- If you want to edit your code but don't want to lose what you have so far you can create a new branch
- When you create a branch off the main branch you are making a copy of main
- You use branches to experiment and make edits before committing them to main

## Branching


'master' branch
Create 'feature' branch from 'master'
Merge 'feature' branch into 'master'
Commit changes
Submit Pull Request
Discuss proposed changes

## Github lingo - Commits

- On GitHub, saved changes are called **commits**
- Each commit has an associated commit message
  - a description explaining why a particular change was made
  - commit messages capture the history of your changes
  - other contributors can understand what you have done and why.

## Exercice 1 - Create a repo

- Create a new repository:
  - Go on your Github page and click on the gree button “New”
  - call this new repository repo-mont2
  - Select: Add a README file
  - Click on Create repository at the bottom

## Exercice 1 - Import locally

- Go on your Github desktop app
- At the top left use the drop down arrow
- Select:
  - Add
  - Clone repository...
  - URL
- Enter the URL of your Github project
  - For instance mine is : https://github.com/forgef/repo-mont2
- At the bottom click on Choose...
  - Locate where you want to put the files
  - Give a name to the folder like repo-mont2

## Exercice 1 - Modify and Commit

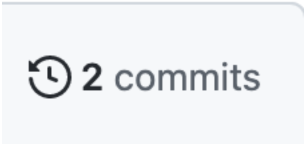
- Locate your README.md file
- This is a markdown file which you can edit from any texteditor (including Latex)
- Go back to your Github desktop app
  - You should now see the changes that were made to your README
- At the bottom left of the app there is a box for you to fill out in which you **must** describe the modifications that you made
  - In sumary type: my first commit
  - In description type: I changed the README using the text editor XXX
- Then click on commit to main

## Exercice 1 - Push

- At the top of the Github app you have a Push origin button
- Click on it
- Go back to your Github page
- Refresh your screen

## Exercice 1 - History

- Above your README file you have the list of the files in your repository and commit messages
  - of course at this stage you only have
- Click on the history button (to the right)



- You can now see the history of the versions of this file
  - the initial and the one you edited
- Go back to the Github desktop app
  - At the top left select history
  - You can also see the old version

## Exerice 1 - Changes on line

- You can now go back to the Github website.
- Locate the edit button (looks like a pencil)
- Edit the README file directly online:
- At the bottom of the README file add “Hello again this is my second edit to my first repository”
- Scroll down to Commit Changes
  - Enter: “my second edit”
  - click on commit