GitHub Desktop

Tugba Bozcaga
MIT

-

Slides at <https://github.com/tugbabozcaga

30 November 2018

What is Version Control?

Demos

What is Version Control?

What is Git 1/2

- Git is a software designed to track the **entire** history of the code of a project.
- Main appeal: facilitates full reproducibility and collaboration.
- Designed originally for software development, it has gained important traction in the research community.
- With GUIs (GitHub Desktop) the learning curve decreased and benefits started to exceed the costs.

Graphical **U**ser **I**nterface

What is Git 2/2

- By code git understands any type of plain text file (`myfile.R`, `myfile.do`, `.tex/.md/.txt/.csv/.etc`).
- Files that are "non-human readable" are called binary files (`myfile.docx`, `myfile.xlxs`, `.pdf/.exe/.dta/.etc`).
- Git can also detect changes in binary files, but it cannot show those changes.

What is Github

Github is a company that provides two services (that we care of):

- A web hosting service for all our files track with git (public free/private $ or free if academic).
- A GUI software (Desktop App) that provides user friendly access to git.

Goal

Goal 1: Keep track of any potentially meaningful modification to your code.

Goal 2: Learn how to collaborate with others using Github.

Conventional Method:

1 - Agree on a naming convention with you co-authors (eg: YYYYMMDDfilename_INITALS).
2 - Begin working from the last saved version (eg: `20180325demo_FH.do`).
3 - At the end of the day, save on a new version (eg: `20180327demo_FH.do`).

Pros: Easy adoption.

Cons: Lots of files for each document, hard to see changes of your own and others, hard to track the time of changes, errors and time costs.

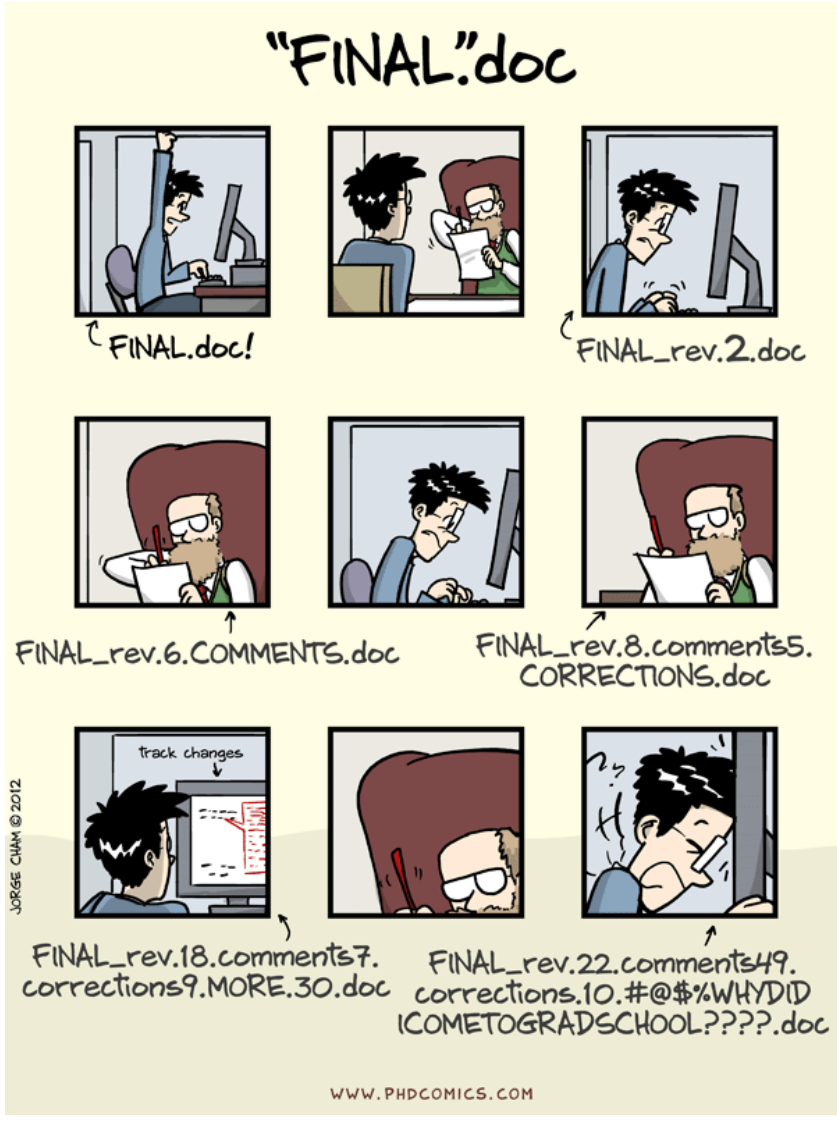
Github:

1 - Name your file `filename` (ideally `01_filename`). 2 - Takes a snapshot of your work every time you complete relevant change. 3 - Updates your entire working folder in the cloud.

Pros: One file per document, track differences across all versions, easily follow when others make a change, works with the cloud.

Cons: Harder adoption.

We want to avoid this situation:



Other reasons to use git

To access a whole new world of knowledge!

- Collaborate in others' open access projects (e.g. R packages).
- Find collaborators for your own projects.

Demos

Three Demos:

1 - **Work on a solo project.**

2 - Copy and work on another person's project.

3 - Collaborate with other people in a project.

Demo #1: We Start in the Cloud

1- Create github.com account and sign in.
2- Let's look at some **repos**.
3- First way to access content: download.
4- What if you want to have your own copy of the repo? **Fork** it!.
5- Now create your own repo. Initiate readme and make some edits.

Demo #1: We move to our local computer

6- Clone the it. Explore the files and location.
7- Create new files, edit. And commit. Edit again, and commit again.
8- Push. Edit on github.com, and pull.
9- Simulate conflict (between local and remote) and start from a fresh copy! 10- For this tutorial, best way to access previous version: explore in github.com and download.

Three Demos:

1 - Simple but instructive.
Review: def repo, github.com, download, clone, destination folder, fork, create repo, commit, push, pull, delete & restart, search repo, download old version.
2 - **Repeat but with a slightly more fun example. Collaborate.**

3 - Repeat with a real-life example.

Demo #2: Branches and collaboration

1- Create a branch from previus repo.
2- Add new content, commit a few times and merge.
3- Go back to main branch (master), observe file, merge.
4- Repeat 1-3 but now replace instead of adding content.

Demo #2: Branches and collaboration

5- Fork repo `github.com/BITSS/test2`, and clone it into your machine.
6- Edit fields of name, and birth date.
7- Save, commit and push.
8- Create your first Pull Request.
9- Let's see if I can manage all those pull requests very quickly.
10- Create a new repo, invite a collaborator, edit, commit, push/pull.

Three Demos:

1 - Simple but instructive.
Review: def repo, github.com, download, clone, destination folder, fork, create repo, commit, push, pull, delete & restart, search repo, download old version.
2 - Repeat but with a slightly more fun example. Collaborate.
Review: All of the above, plus: branch, merge, resolve conflicts, collaborate: same proj, fork model+PR .
3 - **Repeat with a real-life example.**

Demo #3: Look inside a half-way project (and collaborate!)

Description:

- Half baked project, forgoten from a few years.
- Exploratory analysis of publication trends in NBER working paper series. Back then inspired by a paper from DelaVigna and Card.
- Now there is more literature around this: Chari and Goldsmith-Pinkham.
- I will use github to share my work with you, do a little excercise, and invite you to collaborate.

Demo #3: Look inside a half-way project (and collaborate!)

1- Find the following repo: `github.com/fhoces/nber_trends`.
2- Fork it and clone it.
3- Open it in your computer, look around and try to execute some parts.
4- Generate random number like this: `num1 = sample(20000, 1)`.
5- Look the name and (imputed) gender of the author in row `num1`.
6- Create the following line at the end of the script: `verification = (num1, author, gender, correct(1 yes, 0 no) )`.
7- Save, commit, push and create a pull request.
8- Feel free to look around create more contributions if you like. Happy to co-author.

Three Demos:

1 - Simple but instructive.
Review: def repo, github.com, download, clone, destination folder, fork, create repo, commit, push, pull, delete & restart, search repo, download old version.
2 - Repeat but with a slightly more fun example. Collaborate.
Review: All of the above, plus: branch, merge, resolve conflicts, collaborate: same proj, fork model+PR .
3 - Repeat with a real-life example.
Review: All of the above, plus: how does a real-life example looks like.

Now go and explore!

Some good habits:
- Commit often (<1hr)
- Always pull before you start a new session of work. Also good to pull before pushing.
- Think of your remote as the most important set of files. Get used to deleting things in your local machine.

Want to learn more:

- Software Carpentry's step-by-step tutorial (command line).
- Garret Christensen's version of this tutorial.
- Great 20 min intro to Git by Alice Bartlett
- Great 2hr tutorial to Github by Jenny Bryan (git ninja)
- Jenny Bryan's Happy Git; Documentation from Matthew Gentzkow Jesse Shapiro; Karthik Ram's paper on Git for Research