Workshop: GitHub with GIU

Tugba Bozcaga
MIT

30 November 2018

Introduction

What is Version Control

Exercises

Introduction

Slides: https://github.com/tugbabozcaga/GitHub-Tutorial.

Resource: Fernando Hoces's Workshop from Berkeley Initiative for Transparency in Social Sciences.

What is Version Control

What is Git

- A software designed to track the history of the code (known as version control) of a project.
- Facilitates reproducibility and collaboration.
- Designed originally for software development, but widely used in the research community today.
- With GUIs (GitHub Desktop) the learning curve decreased and benefits started to exceed the costs.

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

What is Git

- Git understands any type of plain text file (`myfile.R`, `myfile.do`, `.tex/.md/.txt/.csv/.etc`).
- Files that are "not trackable" are called binary files (`myfile.docx`, `myfile.xlxs`, `.pdf/.exe/.dta/.etc`).
- Git can see the updated versions of binary files, but it cannot track or show the changes made in them.

What is Github

Github provides two services:

- A web hosting service for all our files that we track with git (Public files are free, private files are free only for academics).
- A GUI software (Desktop App) that provides user friendly access to git.

Why to Use

1: To keep track of all the important modifications to your code.

2: To learn how to collaborate with others using Github.

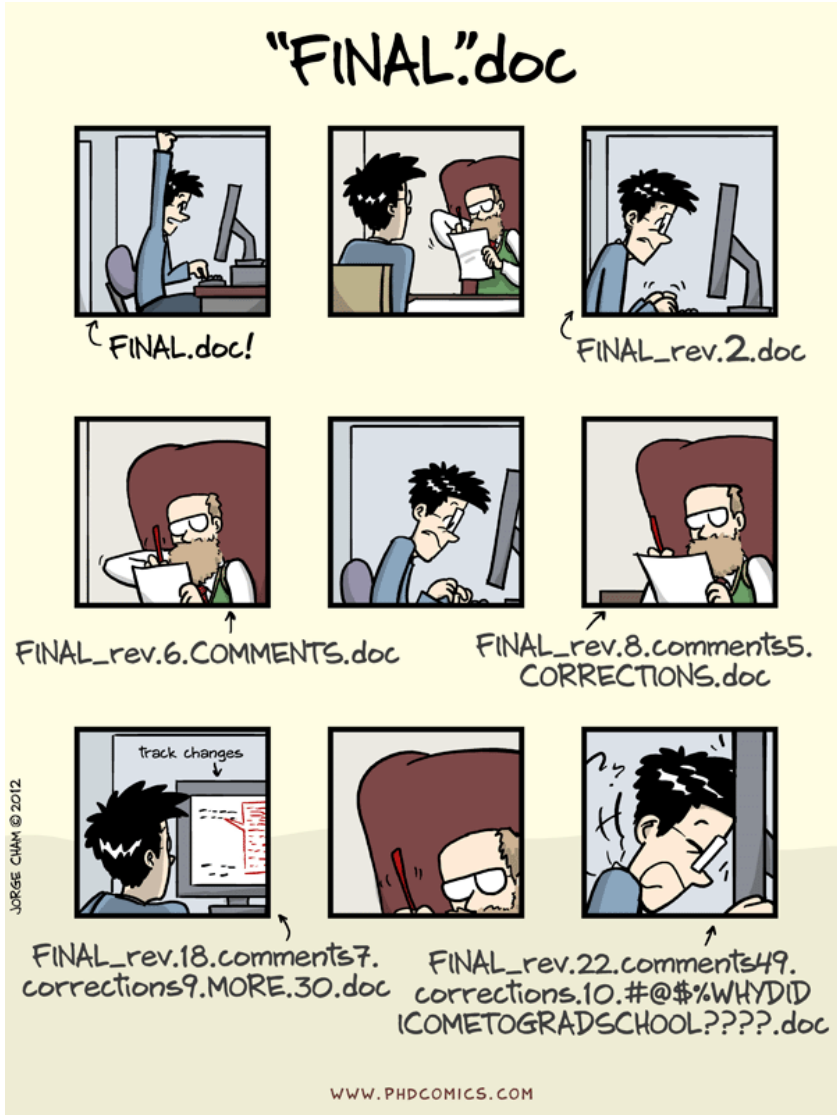
Conventional Method:

1 - Agree on a naming convention with your co-authors (eg: filename_date_INITALS).
2 - Always work from the last saved version (eg: `workshopcode_20181002_OB.do`).
3 - After your changes, save a new version (eg: `workshopcode_20181003_TB.do`).

Easily adopted but:

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.

Example


"FINAL".doc
FINAL.doc!
FINAL_rev.2.doc
FINAL_rev.6.COMMENTS.doc
FINAL_rev.8.comments5.
CORRECTIONS.doc
track changes
FINAL_rev.18.comments7.
corrections9.MORE.30.doc
FINAL_rev.22.comments49.
corrections.10.#@$%WHYDID
ICOMETOGRADSCHOOL????.doc
JORGE CHAM © 2012
WWW.PHDCOMICS.COM

Github

1 - Name your file `filename` (ideally `01_filename`). 2 - Whenever you complete a relevant change and sync, Git takes a snapshot of your work. 3 - Git keeps updating your entire folder in the cloud. Thus, it is visible to your collaborators too.
Harder adoption but:

One file per document, track differences across all versions, easily follow when others make a change, easily follow the time of changes.

Additional Benefits

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

Exercises

Three Exercises

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

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

2b - Send an edit suggestion to the other person's project.

3 - Collaborate with another person in a project.

Exercise #1

We start on Desktop.

- Click **File** and **New Repository**.
- Name it **My-First-Repository** and check "Initialize this repository with a README" (Not required but usually a good idea). Find the location of the new repository folder in your computer.
- Click **Publish repository**. Now your repository is online on Github.com.
- You can visit your online repository by clicking **Repository** and **View on Github** in the menu.

Exercise #1

- Drag any text file to your repository folder.
- Type “My first changes” in your text file.
- Check the GitHub Desktop. You will see the change you just made. Click **Commit** to confirm your change. (You can edit and commit as often as you like, GitHub Desktop will memorize all the changes you committed but they won’t appear online until you push.)
- Click **Push** to see your changes online.

Three Exercises

1 - Work on a solo project.

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

2b - Send an edit suggestion to the other person's project.

3 - Collaborate with another person in a project.

Exercise #2a

We start in the Cloud.

- Sign in your github.com account.
- Find the repo **tugbabozcaga/GitHub-Tutorial**.
- **Fork** the repo (You could also simply clone or download, but fork for this one because we'll send a pull request to the repository owner later.).
- Now click **Clone** and **Open in Desktop**. Now you have downloaded your fork to your desktop too.

Exercise #2a

Now we are on Desktop.

- You will see a window with the button **Clone** in your Github Desktop, click it.
- A folder named **GitHub-Tutorial** has been downloaded to your computer. Find it.
- Open the Birthdays.R file. Type your birthday. Save.
- Go to GitHub Desktop. You will see the change you just made. Click **Commit** to confirm your change.
- **Push**.Now if you visit your own Github-Tutorial fork (youraccount/GitHub-Tutorial), you will see all the changes you made through GitHub Desktop.

Three Exercises

1 - Work on a solo project.

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

2b - **Send an edit suggestion to the other person's project.**

3 - Collaborate with another person in a project.

Exercise #2b

Now you made changes in this other person's project but they only appear in your fork! You want to send her your suggestions/revisions.

- Now that you confirmed that your online GitHub-Tutorial fork reflects all the changes, you can click **Pull Request**.
- Click **Create Pull Request**. Type a short summary explaining your change in the title of the box that emerges, and click **Create Pull Request** again.
- The owner of the repository will click the following options to accept your changes: **View the Pull Request**, **Merge Pull Request**, and **Confirm Merge**. That means now all your changes have been added to the original folder.

Three Exercises

1 - Work on a solo project.

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

2b - Send an edit suggestion to the other person's project.

3 - **Collaborate with another person in a project**.

Exercise #3

Pair up with a neighbor. One of you be A and one be B for this exercise. You can work with your **My-First-Repository** folder.

- A: In the settings tab for A's repository on Github.com (that A published above), add B as a collaborator.
- B: Accept the invitation, and clone A's repository so that you have it in your own computer.

The rest is the same as working on your solo project.

- B: Now make a change, e.g. "Edit 1 by Collab B", commit, and sync (push) the change.
- Switch roles between A & B and repeat.

Tips

- When you collaborate, you may not want to work in the master branch. You can instead create an experimental branch by clicking the **Current Branch** menu, **New Branch**, and then **Create Branch**. This way we'll make sure we can't directly push the changes and can send create a pull request so that person A will see our changes as well. Person A will see the change and merge the pull request.
- If you are collaborating, always sync before you start a new session of work. Also good to sync before pushing.
- You don't have to push every hour, but commit often ($<$1hr)
- Think of your remote as the most important set of files. Get used to deleting things in your local machine.

More resources

- textcolor{blue}{Intro to Git by Alice Bartlett}
- Jenny Bryan's textcolor{blue}{Happy Git} and [textcolor{blue}{Tutorial to Github}] (https://www.rstudio.com/resources/videos/happy-git-and-gihub-for-the-user-tutorial/).
- Karthik Ram's paper textcolor{blue}{Git for Research}.
- Software Carpentry's textcolor{blue}{step-by-step tutorial (command line)}.