Collaborative research with Git

Davor Cubranic and Rick White
Applied Statistics and Data Science Group
UBC Statistics

April 27 2016

Do you have all pre-requisites?

- Github account
- Git and Bash
- up to date R and Rstudio

If not, follow the setup instructions at asda.stat.ubc.ca/workshops/2016-04_ECOSCOPE_topics/setup.html

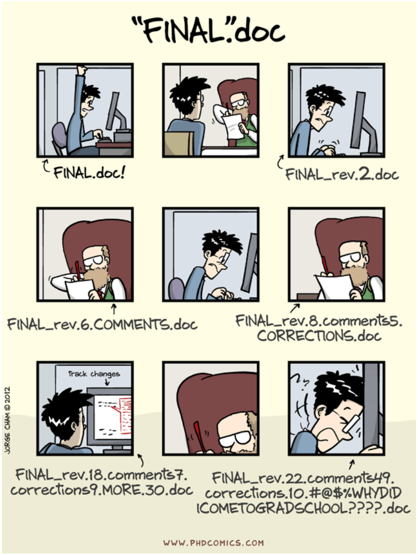
"FINAL".doc
FINAL.doc!
FINAL_rev.2.doc
FINAL_rev.6.COMMENTS.doc
FINAL_rev.8.comments5.CORRECTIONS.doc
track changes
FINAL_rev.18.comments7.corrections9.MORE.30.doc
FINAL_rev.22.comments49.corrections.10.#@$%WHYDIDICOMETOGRADSCHOOL????.doc
JORGE CHAM © 2012
WWW.PHDCOMICS.COM

Figure 1: http://www.phdcomics.com/comics/archive.php?comicid=1531

How to track changes to your files

- nothing: live in the here and now
- date-stamped directories/files
- version control tools

“Automating” version control

Version control tools:

- save the entire history of changes in the project
    - commented, time-stamped, author identified

Benefits of version control

- easy to see what changed when, why, and by whom
- jump back to any point in time
- freedom to experiment
    - try out a crazy idea
    - can always go back to the last known good version
    - keep working on both in parallel
- manage contributions from multiple people

Learning goals

- use Git version control tool in your projects
- collaborate more effectively with Github online service
    - manage simultaneous/conflicting changes by various contributors
    - use Github Issues for tracking TODOs and focus discussion

Version control with Git

What is Git?

- **a** version control system
- a **distributed** version control system
- originally developed to manage source code for Linux

Why use Git and not *fill in the blank*?

- it is very fast, even for projects with hundreds of files and years of history
- doesn't require access to a server
- extremely popular, no strong reason to pick anything else (unless required)

Before we start

Tell Git who you are:

In RStudio, go to “Tools” -> “Shell…” and enter:

```
git config --global user.name "My Name"
git config --global user.email "me@email.com"
```

Basic use

- create a new project
    - check “create a Git repository”
- see the “Git” tab
    - shows two files
    - status: “? ?”
- click in the “Staged” column
- commit

Reviewing changes

- create a file Readme.md
- add and commit
- view history tab

Terminology

Repository	a store of contents and history of all files in a project
Commit	a permanent snapshot of the state of the repository at a point in time • annotated with author, time, and comment • linked with a preceding (“parent”) commit
Staging	files marked for inclusion in the next commit • as they were *at the time they were staged!!*

Exercise

1. Make more changes to Readme.md
2. Create a new file (R script or R/markdown document)
3. Add and commit

Reviewing the history

- switch to the Git history view (“clock” icon in the Git tab)
- list of commits
- each shows what it changed (“diff”)
- commit “name” (unique SHA-1 fingerprint)

Undoing changes

If you:

- change your mind about your edits
- accidentally delete a file

Then:

- before committing:
  - right-click in Git tab and “Revert” to throw away the changes; or
  - in the “diff” view, click on “discard chunk” or “discard line”
- after committing: go back to a previous commit
  - must use shell: `git revert`

Branching and merging

- use branches to test out new features without breaking the working code
    - must use shell: `git branch branch-name`
    - can switch branches in RStudio's Git tab
- when the work in a branch is done, merge it back into the main ("master") branch
    - must use shell: `git merge branch-name`

Exercise

- create a new branch “develop”
- make some changes to the project (edit, delete, or create new files)
- add and commit
- switch to the “master” branch
- compare project contents
- merge the “develop” branch
- what happens?

Merge conflicts

Git tries its best to merge changes from different branches, but sometimes it fails (“merge conflict”):

- switch to the develop branch
- edit and commit
- switch to the master branch
- change the same lines and commit
- try to merge “develop”

Resolving merge conflicts

- boths sets of conflicting lines are preserved in the file
    - look for markers <<<<<<, =======, and >>>>>>
- edit the file until it looks the way you want it (removing the markers)
- add and commit
- conflict solved!!

Terminology

Branch	a named sequence of commits kept separate
Merge	including changes from one branch into another
Conflict	parallel changes to same section(s) of a file that can't be automatically merged

Collaboration with Git and Github

What is Github?

- service that hosts your repository online
- web interface for Git
- features to help work with contributors/collaborators

Why use Github and not *file in the blank*?

- nice graphical user interface for Git
- provides a copy of the repository that is easily accessed by others
- a variety of add-ons to facilitate collaboration
- *huge* community

Github repository

- go to Github.com and sign-in
- click “Create a new repo”
- give it a name and description
- click “Create”

Fork a repository

- view someone else’s repository
- click “Fork”
- you get a copy of the repository under your account

Issues

- bugs to fix
- features to add
- any “todo”
- mark who is responsible by “assigning”
- use “labels” for: milestones, type of todo, etc.

Pull requests

- Github tracks differences between your fork and the original repository
- can request that changes be included in the original

Exercise

Use the handout to work through the exercise. Work in pairs.

Other Github goodies

- Wiki pages for documentation
- build a web site from a repo
- private repositories
- organizations