Motivation: Version Control with Git as a Learning Objective in Statistics Courses

Matthew Beckman
Penn State University

Month DD, YYYY
Location

Outline

From the skeptics. . .

- Seriously? *Another* learning objective to squeeze in??
- *Which* statistics course?
- Can't they just pick it up when they need it?
- In fact, *do* they even need it?

Who Cares?:

> Bollen et al (2015) Advisory committee to NSF: Reproducibility describes "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."

- (2014) ASA Curriculum Guidelines for Undergraduate programs in Statistical Science
  - reproducibility
- (2017) ASA Recommendations to Funding Agencies for Supporting Reproducible Research
  - "Use of version control for all (collaborative or individual) code development"
- (2016) Park City Report–Curriculum Guidelines for Undergraduate programs in Data Science
  - discusses both reproducibility & version control
- CS education calls for VC in the curriculum (Haaranen & Lehtinen, 2015; Zagalsky et al., 2015)
- 2017 Kaggle Study
  - almost 17,000 responses
  - over a third reported using a VC tool
  - 58% of those using VC use Git

Version control

```
- maintains the evolution of the project
- collaboration among multiple contributors
- self-collaboration with multiple computers
- branches permit parallel development
```

- Reproducibility ≠ Version Control
  - A standalone RMarkdown doc can be reproducible
  - Version control can manage a group a files that are NOT reproducible

but the investment in good habits with a professional workflow pays off enormously in the long run...

- Ethical practice
  - Any analysis may require hundreds of tiny decisions
  - Work products often intended for audience without technical expertise to scrutinize those decisions
- Academic Preparedness
  - programming is a collaborative sport
  - effective entry point for research participation
- Industry Preparedness
  - programming is a collaborative sport
  - quite common to refresh standard reports

We often lump together Reproducbility and VC as if they're one in the same and speak of Git(Hub) as a panacea. They aren't and it isn't, but the investment in good habits with a professional workflow pays off enormously in the long run...

- 2014 ASA Curriculum Guidelines push for reproducibility
- CS education calls for VC in the curriculum (Haaranen & Lehtinen, 2015; Zagalsky et al., 2015)
- 2017 Kaggle Study
  - [aside] **a third reported using a VC tool!**
  - I'll bet there's *something* in one of my classes that fewer than a third of my students are likely to use in their first 3 years on the job... (anyone ever teach Latin Square designs?)

Q & A

Jo's video question

Q1. What is Git telling me to do? "I don't understand what 'stashing' means

A1. Merge conflicts are the way Git says "I need a human!"

I don't claim to be a particularly advanced Git user myself, but your workaround is perfectly reasonable (it's what I would have done anyway) and ultimately identical to "Git" way if you take a trip to the Terminal in this case. Git **can't** merge binary files (e.g., PDFs). Eventually, you'll just end up deleting it one way or the other

stashing preserves your work in progress (WIP) so you can choose to (a) move on without it (b) come back to it later (c) restore it on a new branch

https://git-scm.com/book/en/v2/Git-Tools-Stashing-and-Cleaning

`git stash` # saves WIP–work in progress somewhere else so you can restore later `git stash list` # air the dirty laundry–show all stashed WIP (i.e., loose ends) `git stash apply` # restore to stashed state–FAILS in this case due to the original issue `git stash branch` # creates a new branch to restore the WIP again

The branch just moves the problem, and doesn't solve it because git **can't** merge binary files (e.g., PDFs). Eventually, you'll just end up deleting it

`git clean` # deletes any stashed WIP `git checkout` # takes `--ours` or `--theirs` options to select one side or the other

Q2. workaround–delete the PDF (burn it down!)

A2. Once PDF is deleted, there's no longer a local file in conflict with the remote version. As far as Git is concerned for that specific PDF, it's a clean slate.