QSS20: Modern Statistical Computing

Unit 04: Workflow tools

Goal for today's session

- **Housekeeping:** prelim discussion of problem set one feedback (longer review next Monday); upcoming assignments
- Workbook with three concepts from pset one that came up in OH: subsetting rows using logical indicators; proportion as mean of a True/False indicator; writing user-defined functions
- Basic command line syntax
- Git/GitHub

Goal for today's session

- **Housekeeping: prelim discussion of problem set one feedback (longer review next Monday); upcoming assignments**
- Workbook with three concepts from pset one that came up in OH: subsetting rows using logical indicators; proportion as mean of a True/False indicator; writing user-defined functions
- Basic command line syntax
- Git/GitHub

Problem set 1 feedback (thus far; 21% response rate)

- **Course pace:** 66% said "just right"; 33% said too fast
- **Pset 1 difficulty:** 50% said "just right"; 50% said too difficult. Improvements:
    - **Already implemented in pset two (thanks to early TA feedback!):**
        - **Explicitly note concepts and resources to minimize amount needed to Google:** added notes in each question that notes which specific concepts from class a question tests and links to the slides/notebooks with relevant code
        - **Hints on output:** pset2 hints on output so your goal is just to reproduce
        - **Collaboration**: self-chosen partners (2 max) or partner pool - link partner pool
        - Separated questions into Part A, B... with separate cells so easier to break down/debug— also free to add as many cells as you want!
    - **Not yet implemented:** pset working time during class (tradeoffs w/ material coverage); CS terminology word bank (love that idea! maybe a google doc or running github issue?)
- Discuss concerning comment about course oversight/removal from QSS curriculum

Calibrating level of challenge

1. **DataCamp**: meant as gentle intro before pset challenges; not realistic for entry-level data science jobs; provides a lot of handholding in terms of noting (1) exactly which commands to use; (2) helper code; (3) very simplified/cleaned data
2. **Real-world data science:** more difficult than the problem set; would be asked "hey, did this policy reduce or widen disparities" and start with a blank notebook and be 100% reliant on google/stackoverflow
    - **Translating question into concrete approach:** define disparities (charges, incarceration or not, sentencing conditional on incarceration); find which variables measure that; deal with duplicates
    - **Data cleaning without scaffolding:** recognizing the errors in the datetime and that `errors_coerce = True` would set a lot of valuable data to missing; further deduplication of judge names; investigating `PROMIS CONVERSION` (eg coding that to missing)
    - A lot of these things won't throw errors if you run an analysis without fixing but will lead to flawed results/incorrect policy conclusions

Upcoming assignments

- **DataCamp on merging in pandas:** due Monday by class; reminder that you can skip all DataCamps and reallocate that 5% to psets
- **Problem set 2:** due next Friday 01.28 11:59 PM; will not be an extension until that sunday but feel free to use late days (but need to coordinate within partner)
- **Grades for pset 1:** if you submitted on time, hopefully by next Monday

Concepts in problem set two

Problem set question	**Concepts**
1.1 Filter to incarceration and construct a sentence length variable	Row filtering; loop or function
1.2 Focusing on specific judge and narcotics offenses, match each of the defendant's to a group of other defendants who were (1) sentenced by same judge; (2) have same age and gender; (3) have different race; investigate disparities	Row filtering; loop or function

GitHub issues vs. Slack channel vs. Slack DMs

- Slack channel: general workflow questions (eg deadline)
- GitHub issues: specific questions (same no code rule applies; shifted to improve organization of responses)
    - **How do I post?** Google doc with software guides
- Slack DM: to help unclog office hours; for syntax debugging, can DM one of us with code and screenshot of errors; not instant response; try to follow Slack on call schedule posted on Canvas

Goal for today's session

- Housekeeping: prelim discussion of problem set one feedback (longer review next Monday); upcoming assignments
- **Workbook with three concepts from pset one that came up in OH: subsetting rows using logical indicators; proportion as mean of a True/False indicator; writing user-defined functions**
- Basic command line syntax
- Git/GitHub

Open up notebook we were using for functions

Pause for practice

Add cells to 02_loopsfunctions.ipynb

Goal for today's session

- Housekeeping: prelim discussion of problem set one feedback (longer review next Monday); upcoming assignments
- Workbook with three concepts from pset one that came up in OH: subsetting rows using logical indicators; proportion as mean of a True/False indicator; writing user-defined functions
- **Basic command line syntax**
- Git/GitHub

Why are we covering this?

- **Easiest way to interface with Git/GitHub:** as we'll discuss next, Git/GitHub have a graphical user interface (GUI), or a way to go to a website and point/click, but that defeats a lot of the purpose
- **Moving files around on jupyter hub**
- **TBD: interacting with high-performance clusters/long-running jobs:** a lot of what we'll be doing is code written in jupyter notebooks (.ipynb) that runs relatively quickly; if we have time to cover high-performance computing, running `.py`

Where is the "command line" or what's a terminal?

- Mac default one- open up spotlight and search for terminal
- Windows terminal emulators - see list here https://rebeccajohnson88.github.io/qss20/docs/software_setup.html

First set of commands: navigating around directory structure

1. Where am I?
   ```
   pwd
   ```
2. How do I navigate to folder *foldername*?
   ```
   cd foldername
   ```
3. I'm lost; how do I get back to the home directory?
   ```
   cd
   ```
4. How do I make a new directory with name *foldername*?
   ```
   mkdir foldername
   ```
5. What files and directories are in this directory? (many more sorting options here: https://man7.org/linux/man-pages/man1/ls.1.html)
   ```
   ls
   ls -t
   ```
6. How do I navigate "up one level" in the dir structure?
   ```
   cd ../
   ```

Activity (on your terminal/terminal emulator)

1. Find your terminal
2. Navigate to your Desktop folder
3. Make a new folder called `qss20_clfolder`
4. Within that folder, make another subfolder called `sub`
5. Enter that subfolder and list its contents (should be empty)
6. Navigate back up to `qss20_clfolder` without typing its full pathname

Second set of commands: moving stuff around

1. Create an empty file (rarer but just for this exercise)
   ```
   touch examplefile.txt
   ```
2. Copy a specific file in same directory (more manual)
   ```
   cp examplefile.txt examplefile2.txt
   ```
3. Copy a specific file in same directory and add prefix (more auto):
   ```
   for file in examplefile.txt; do cp "$file" "copy_$file"; done;
   ```
4. Move a file to a specific location (removes the copy from its orig location; root path differs for you)
   ```
   mv copy_examplefile.txt /Users/rebeccajohnson/Desktop/qss20_clfolder/
   ```
5. Move a file "down" a level in a directory
   ```
   mv copy_examplefile.txt sub/
   ```
6. Move a file "up" one level
   ```
   mv copy_examplefile.txt ../
   ```
7. Up two levels:
   ```
   ../../
   ```

Third set of commands: deleting

1. Delete a file

   `rm examplefile.txt`

2. Delete a directory

   `rm -R examplefile.txt`

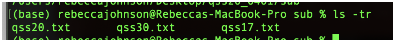
3. Delete all files with a given extension (example deleting all pngs; can use with any extension)

   `rm *.png`

4. Delete all files with a specific pattern (example deleting all files that begin with phrase testing)

   `rm testing*`

5. Can do more advanced regex- eg, deleting all files besides the qss20 one in this dir



```
(base) rebeccajohnson@Rebeccas-MacBook-Pro sub % ls -tr
qss20.txt       qss30.txt       qss17.txt
(base) rebeccajohnson@Rebeccas-MacBook-Pro sub %
```

   `find sub/ -name 'qss[1|3][7|0].txt' -delete`

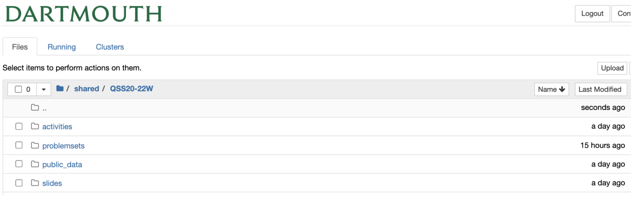
Activity (on your terminal/terminal emulator)

1. Delete the `sub` directory in `qss20_clfolder`
2. Use touch to create the following two files in the main qss20_clfolder: `00_load.py 01_clean.py`
3. Create a subdirectory in that main directory called `code`
4. Move those files to the `code` subdirectory without writing out their full names
5. Copy the `01_clean.py` into the same directory and name it `01_clean_step1.py`
6. Remove all files in that directory with clean in the name

Introducing jupyter hub (jhub) and applying these commands

Jhub: cloud server similar to CoLab; each time it restarts it pulls latest materials from our `qss20_slides_activities` repo; way to access materials once we move away from Canvas posting

1. Navigate to `https://jhub.dartmouth.edu` and click on QSS20 option
2. Shared course materials (slides; in-class activities) are in the following read-only folder (shared/QSS20-22W/):



Select items to perform actions on them.

/ shared / QSS20-22W

..	seconds ago
activities	a day ago
problemsets	15 hours ago
public_data	a day ago
slides	a day ago

3. In the main directory (one level up from shared or the folder icon), create a folder to store editable files: `qss20_mywork`

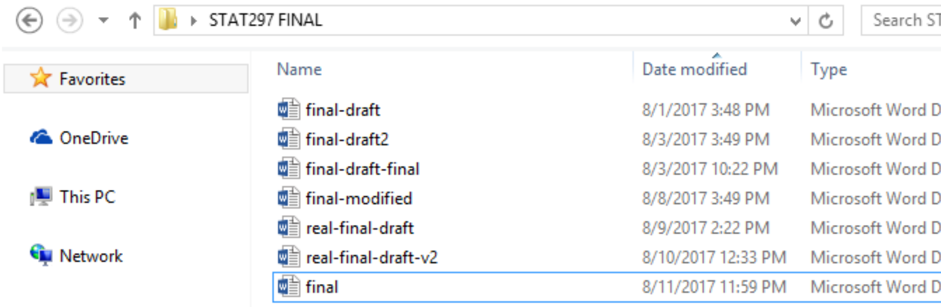
Activity (on jhub)

1. Navigate to the terminal via New $\Longrightarrow$ Terminal
2. If you haven't already, use `mkdir` to create a new directory `qss20_mywork`
3. Copy the following file from "shared/QSS20-22W/activities/w22_activities" into that directory: `00_classquestions.ipynb` (if it's not showing up go to control panel and restart kernel)
4. Rename that file with your netid as a suffix before the .ipynb
5. Practice editing

Goal for today's session

- Housekeeping: prelim discussion of problem set one feedback (longer review next Monday); upcoming assignments
- Workbook with three concepts from pset one that came up in OH: subsetting rows using logical indicators; proportion as mean of a True/False indicator; writing user-defined functions
- Basic command line syntax
- Git/GitHub

Motivation for Git/GitHub



Name	Date modified	Type
final-draft	8/1/2017 3:48 PM	Microsoft Word D
final-draft2	8/3/2017 3:49 PM	Microsoft Word D
final-draft-final	8/3/2017 10:22 PM	Microsoft Word D
final-modified	8/8/2017 3:49 PM	Microsoft Word D
real-final-draft	8/9/2017 2:22 PM	Microsoft Word D
real-final-draft-v2	8/10/2017 12:33 PM	Microsoft Word D
final	8/11/2017 11:59 PM	Microsoft Word D

Source: SMAC group

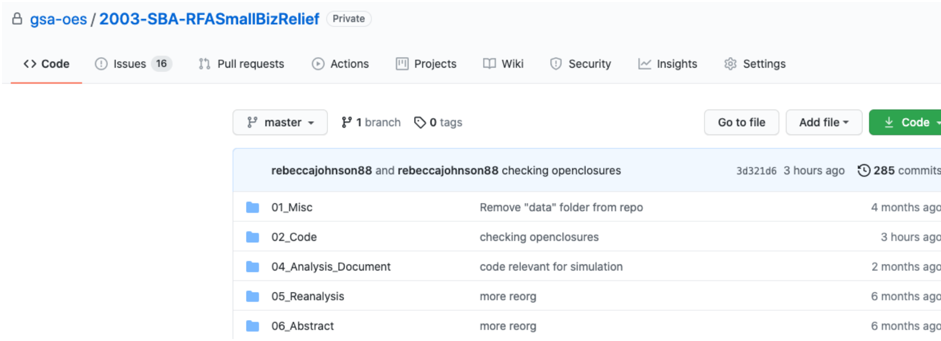
What is Git?

- Set of command line tools for version control (aka avoid finalfinal, finalrealthistime, etc.)
- “Distributed,” or means that files/code, rather than only stored one place centrally, can be stored on all collaborators’ machines

What is GitHub?

- Web-based repository for code that utilizes `git` version control system (VCS) for tracking changes
- Has additional features useful for collaboration, some of which we'll review today (repos; issues; push/pulling recent changes) and others of which we'll review as the course progresses (branches; pull requests)
- Why GitHub rather than Dropbox/google drive?
    - Explicit features that help with simultaneous editing of the same file
    - Public-facing record, or a portfolio of code/work (if you make it public)
    - Ways to comment on and have discussions about code specifically through the interface

Example repo: private repo

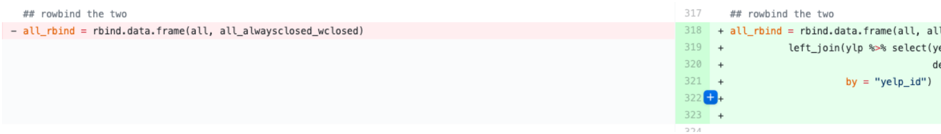


rebeccajohnson88 and rebeccajohnson88 checking openclosures 3d321d6 3 hours ago 285 commits

01_Misc	Remove "data" folder from repo	4 months ago
02_Code	checking openclosures	3 hours ago
04_Analysis_Document	code relevant for simulation	2 months ago
05_Reanalysis	more reorg	6 months ago
06_Abstract	more reorg	6 months ago

If you go to the url, get 404 error unless you're added as a collaborator: https://github.com/gsa-oes/2003-SBA-RFASmallBizRelief

Example: tracked changes in code when you "push" updated version



```
   ## rowbind the two
 - all_rbind = rbind.data.frame(all, all_alwaysclosed_wclosed)
```

```
   ## rowbind the two
 + all_rbind = rbind.data.frame(all, all
 +          left_join(ylp %>% select(ye
 +                                     de
 +                  by = "yelp_id")
 +
 +
```

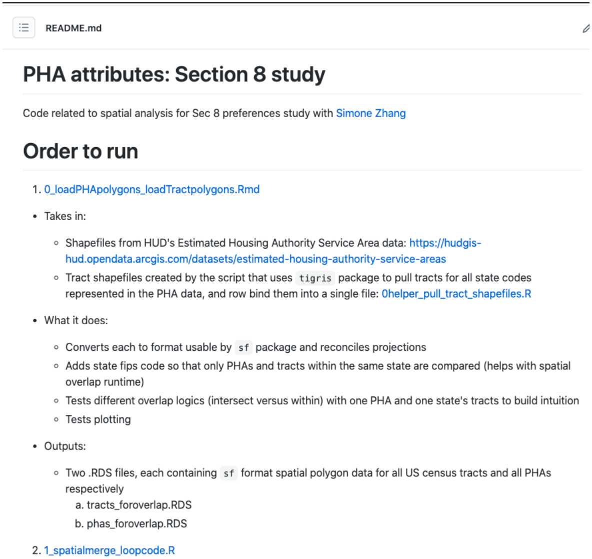
Example repo: public repo

Look familiar?

https://github.com/rebeccajohnson88/qss20_slides_activities

Ingredients of a repo: README

Should be more informative than the above example, e.g.:



README.md

PHA attributes: Section 8 study

Code related to spatial analysis for Sec 8 preferences study with Simone Zhang

Order to run

1. 0_loadPHApolygons_loadTractpolygons.Rmd

- Takes in:
  - Shapefiles from HUD's Estimated Housing Authority Service Area data: https://hudgis-hud.opendata.arcgis.com/datasets/estimated-housing-authority-service-areas
  - Tract shapefiles created by the script that uses `tigris` package to pull tracts for all state codes represented in the PHA data, and row bind them into a single file: 0helper_pull_tract_shapefiles.R
- What it does:
  - Converts each to format usable by `sf` package and reconciles projections
  - Adds state fips code so that only PHAs and tracts within the same state are compared (helps with spatial overlap runtime)
  - Tests different overlap logics (intersect versus within) with one PHA and one state's tracts to build intuition
  - Tests plotting
- Outputs:
  - Two .RDS files, each containing `sf` format spatial polygon data for all US census tracts and all PHAs respectively
    a. tracts_foroverlap.RDS
    b. phas_foroverlap.RDS

2. 1_spatialmerge_loopcode.R

Ingredients of a repo: directories

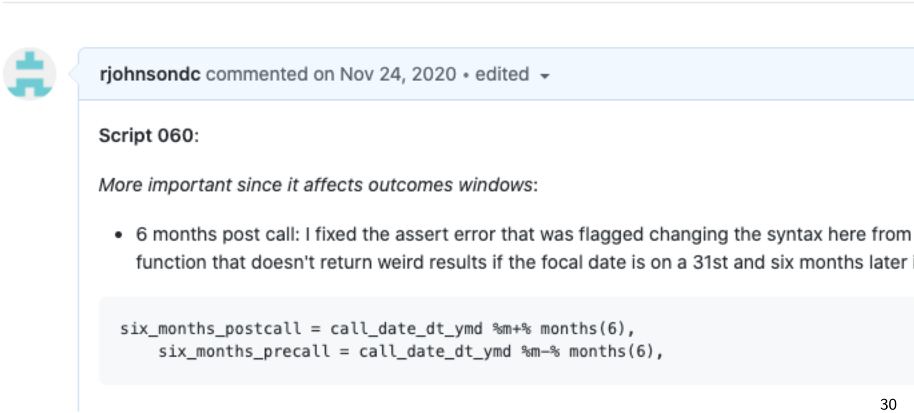
Command line syntax in previous slide is useful for org/reorg. For our class, we'll generally have two directories:

1. code/ (with subdir for tasks)
2. output/ (with subdir for tables versus figs)

Depending on the context, you *may* store data, but (1) GitHub has file size limits, and (2) sensitive data should generally not be put in a repo, even if the repo is private (instead, read directly directly from its source or have download instructions)

Ingredients of a repo: issues

- Can assign to specific collaborators or leave as a "note to self" to look back at something
- Can use checklist features
- Can include code excerpts
- Easy to link to a specific commit (change to code)



rjohnsondc commented on Nov 24, 2020 • edited

Script 060:

More important since it affects outcomes windows:

- 6 months post call: I fixed the assert error that was flagged changing the syntax here from
function that doesn't return weird results if the focal date is on a 31st and six months later i

```
six_months_postcall = call_date_dt_ymd %m+% months(6),
    six_months_precall = call_date_dt_ymd %m-% months(6),
```

General steps in workflow

1. Create or clone a repository to track
2. Make changes to code or other files
3. **Commit** changes: tells the computer to “save” the changes
4. **Push** changes: tells the computer to push those saved changes to github (if file exists already, will overwrite file, but all previous versions of that file are accessible/retrievable)

Create a new repository: instructions

- On GitHub.com: new
- Enter a name (for command line reasons, avoid spaces)
- Give a brief description
- Initialize with a readme
- Add a .gitignore (basically residual files you dont want in repo)
- Select a license

Contribute to a repository

1. Clone repo
2. Edit files
3. Send changes to GitHub (all; would use with caution)

```
git status
git add
git commit -m "this is what i changed"
git push
```

4. Send changes to Github (specific files)

```
git status
git add specificfile.ipynb
git commit specificfile.ipynb -m "this is what i changed"
git push
```

5. Send changes to GitHub (files of a given type; eg you created a bunch of figures that you want to push)

```
git status
git add *png
git commit *png -m "new figs"
git push
```

Focusing on first step: how to clone

1. Open your local terminal and navigate to where you want the repo's files to be stored
2. Go to GitHub.com and go to "Code" button to find the name of the repo
3. Type the following command to clone (reponame.git will be the name of the url you copy/pasted)

```
git clone reponame.git
```

Activity

1. Create a new private repo using the website and instructions on slide 24; name it `qss20_w22_assignments`; add me (rebeccajohnson88) as a collaborator
2. Clone the repo locally using your terminal/terminal emulator
3. Create a `code/` subdirectory
4. Create a `output/` subdirectory
5. Within the code/ subdirectory, move a file you have from another directory to that directory (eg .py, .R, .ipynb) or use `touch` to create blank file
6. Within the output/ subdirectory, use `touch` to create a blank file
7. Push the changes to the code subdirectory
8. Push the changes to the output subdirectory
9. Using the GitHub website, edit the README to link to those changes
10. Assign me an issue
11. Make another change to a file locally (e.g., could edit the text file or add a comment to the code file) and try pushing. You should receive an error if you edited the README non-locally. Try to diagnose by googling, fix, and re-push.

For that last step...

I SEE THAT YOU COMMIT YOUR CODE WITHOUT DOING A PULL FROM GIT TO GET THE LATEST UPDATES
I ALSO LIKE TO LIVE DANGEROUSLY

Additional things we may cover in future session

- **Storing your credentials**
- **Tools for more collaborative coding:** branching and pull requests
- **Options to reverse changes**
- **Slightly different cloning structure on jhub**