QSS20: Modern Statistical Computing

Session 02: Workflow tools

Goal for today's session

- Some course housekeeping
- Basic command line syntax
- Git/GitHub
- LaTeX/Overleaf

Goal for today's session

- **Some course housekeeping**
- Basic command line syntax
- Git/GitHub
- LaTeX/Overleaf

Visit from two public-interest lawyers from our SIP Partner—Texas RioGrande Legal Aid—around 6 pm, so pause at that point; might have some overflow into Tuesday

From last time...

- See Canvas announcement re: which DataCamp modules I'm checking
- Any questions on office hours?
- As you work through assignment due by Tuesday class, try to put at least 1 question here or examples of solutions you found confusing or surprising (counts towards team player portion of grade): https://docs.google.com/document/d/1gYrxrmJcINcNIrs-2tFN8dqLImbvav_VZ6_vA-KiBEM/edit?usp=sharing
- Replace your avatar in Slack so we no longer have a sea of these :)

Brainstorming questions for SIP lawyers

- Skim this practicum doc: https://docs.google.com/document/d/1bmmztzKQ2R_ltL-EYkATfBGeYoZJNIwj0HWcBI6E3BY/edit?usp=sharing
- At the top, add some questions

Goal for today's session

- Some course housekeeping
- **Basic command line syntax**
- Git/GitHub
- LaTeX/Overleaf

Why are we reviewing 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
- **Later, 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; later, executing scripts that might take several hours to run; command line syntax useful

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 (local)

1. Find your terminal
2. Navigate to your Desktop folder
3. Make a new folder called `qss20_0401`
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_0401` 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_0401/
   ```
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`

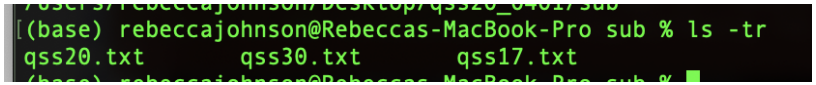
3. Delete all files with a given extension (example deleting all pngs; can use with any extension)

   `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



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

Activity (local)

1. Delete the `sub` directory in `qss20_0401`
2. Use touch to create the following two files in the main qss20_0401: `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

Activity (on jhub)

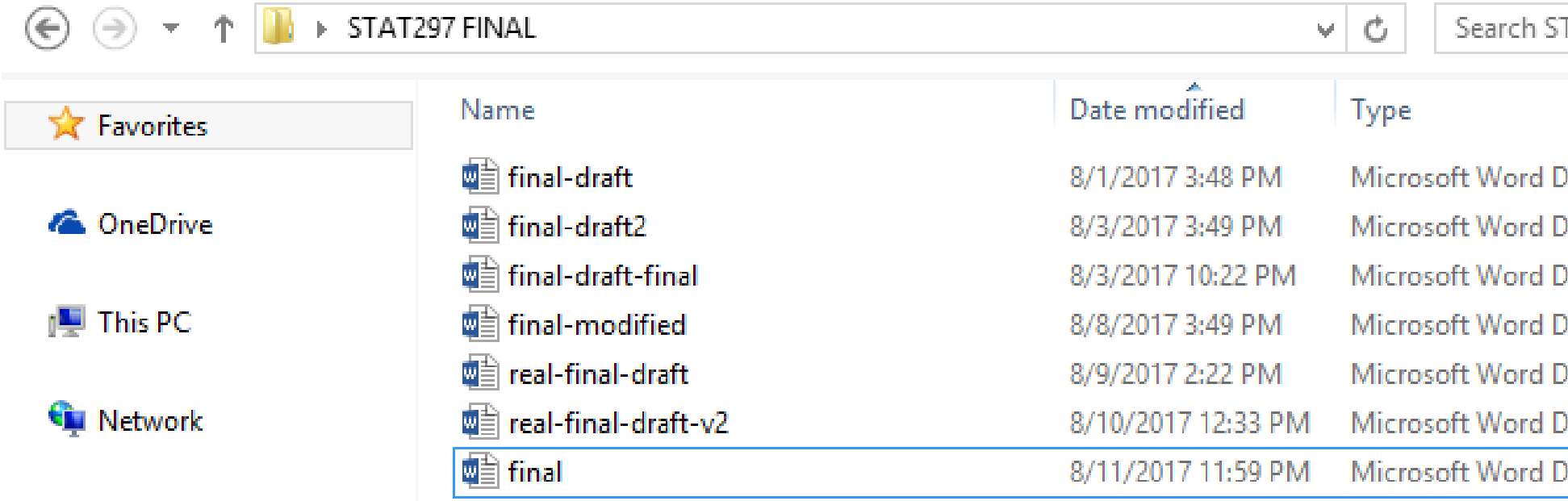
1. Navigate to the terminal via New $\implies$ Terminal
2. If you haven't already, create a new directory `qss20_mywork`
3. Copy the following file from "shared/qss20/activities" into that directory: `00_latex_output_examples.ipynb` (if it's not showing up go to control panel and restart kernel)
4. Rename that file with your netid as a suffix before the .ipynb

Goal for today's session

- Some course housekeeping
- Basic command line syntax
- **Git/GitHub***
- LaTeX/Overleaf

*Some slides and activity adapted from Ryan Moore AU Winter Data Science session.

Motivation for Git/GitHub



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

gsa-oes / **2003-SBA-RFASmallBizRelief** Private

Code | Issues 16 | Pull requests | Actions | Projects | Wiki | Security | Insights

master | 1 branch | 0 tags

rebeccajohnson88 and **rebeccajohnson88** checking openclosures

01_Misc	Remove "data" folder from repo
02_Code	checking openclosures
04_Analysis_Document	code relevant for simulation
05_Reanalysis	more reorg
06_Abstract	more reorg
07_Presentation	fig updates
08_packages/renv	renv implementation
.Rprofile	renv implementation

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 ßave" 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

- 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

Contributing 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 -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 -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_s21_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)
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...

Additional things we'll cover in future session

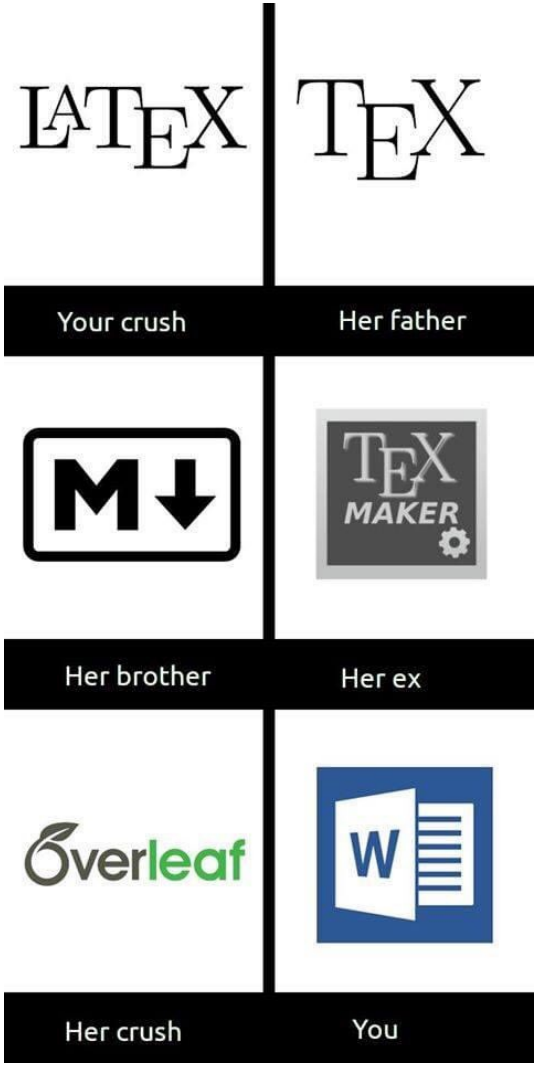
- **Storing your credentials**
- **Tools for more collaborative coding:** branching and pull requests
- **Options to reverse changes**
- **Slightly different cloning structure on jhub**

Goal for today's session

- Some course housekeeping
- Basic command line syntax
- Git/GitHub
- **LaTeX/Overleaf**

Overview before activity

- LaTeX: typesetting language
- As discussed in , can work with locally using things like TexMaker, etc.
- Here, we'll be interacting with it via Overleaf, which is similar to Google docs but for LaTeX and facilitates collaboration/easy(or easier...) troubleshooting of compile errors



Non-exhaustive list of things that can cause compilation errors

- Underscores or certain special characteristics without an "escape" before them– eg:

```
## causes error due to underscore without escape
The file is called: file_here.R
## works
The file is called: file\_here.R
## comments out rest of code after percent symbol
This increased by 5%
## works
This increased by 5\%
```

- Start entering math mode but fail to exit it, e.g.

```
## causes errors
We calculate fraction as $\dfrac{5}{10} and then do...
## works
We calculate fraction as $\dfrac{5}{10}$ and then do
```

"Environments", or ways to go beyond standard text

- Itemized list

```
\begin{itemize}
\item First item...
\item
\end{itemize}
```

- Numbered list

```
\begin{enumerate}
\item First item...
\item
\end{enumerate}
```

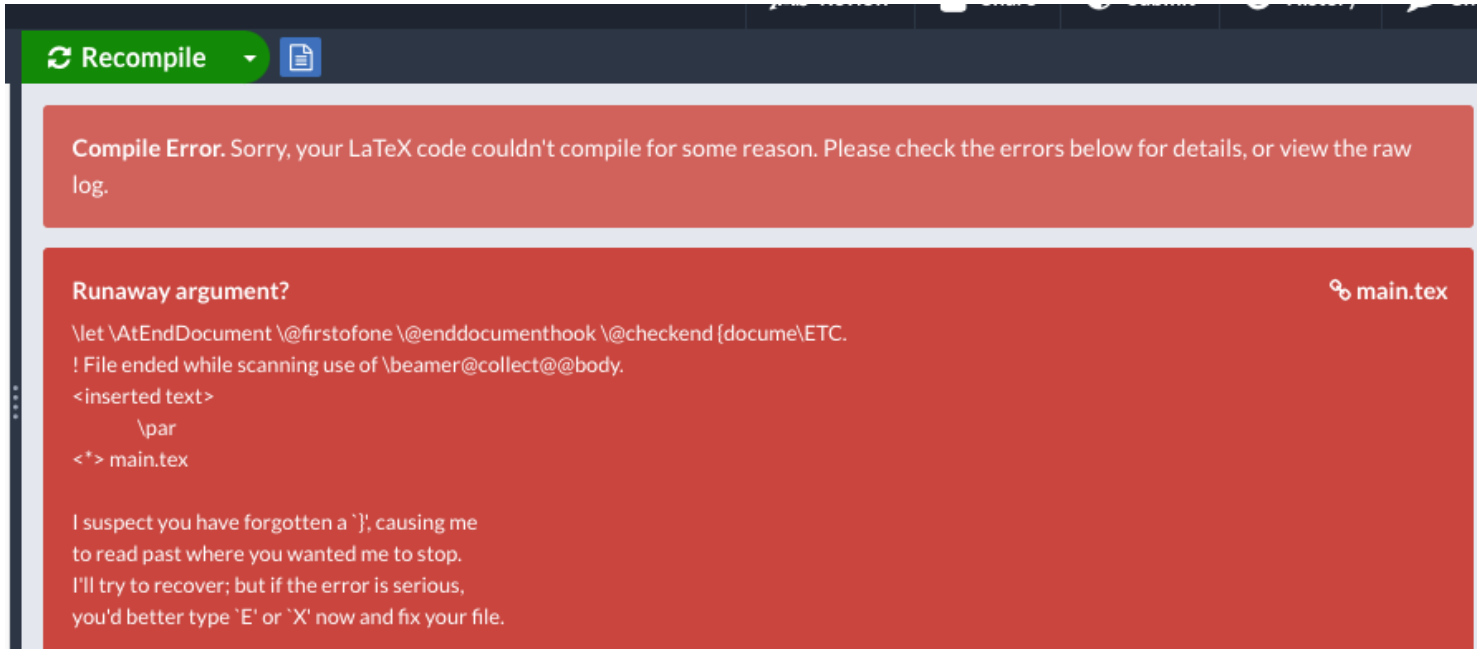
- Figure

```
\begin{figure}
\caption{my caption}
\label{fig:myfig}
\includegraphics[scale = 0.5]{example_graphic.png}
\end{figure}
```

Leads to another set of compilation errors

- Runaway argument or forgotten end group
- Usually means you began an environment but forgot to end it; can happen with long tables, deeply nested lists, etc. where easy to lose track

Example:



Compilation errors

- Common w/ complicated docs
- Ways to address beyond googling: try to recompile relatively frequently since especially on Overleaf, error messages are not always the most informative w.r.t. line numbers

Other useful commands

```
## create a numbered section and give it a label to cross-ref
\section{This is my section outlining disparities}
\label{sec:disparities}

## reference a section in text
In Section \ref{sec:disparities} I discuss...

## reference a table or fig in text
In Table \ref{tab:tabname}, I show why Figure \ref{fig:myfig} shows

## stop a table or figure from going into the next section
## (in addition to stuff at the start of the \begin{table} command
\FloatBarrier
```

Activity

- Visit the example Overleaf doc here: https://www.overleaf.com/9393846375skwbrkgvtkkb
- Copy into your overleaf account and rename (I may need to add you explicitly)
- We'll go over the example part of the activity in shared/qss20/activities/ before I break you into groups to complete the interactive part: `00_latex_output_examples.ipynb`