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`

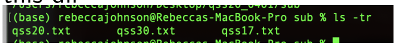
3. Delete all files with a given extension (example deleting all pngs; can use with any extension)

   `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



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

`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 $\Longrightarrow$ 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 (revised 04.06)

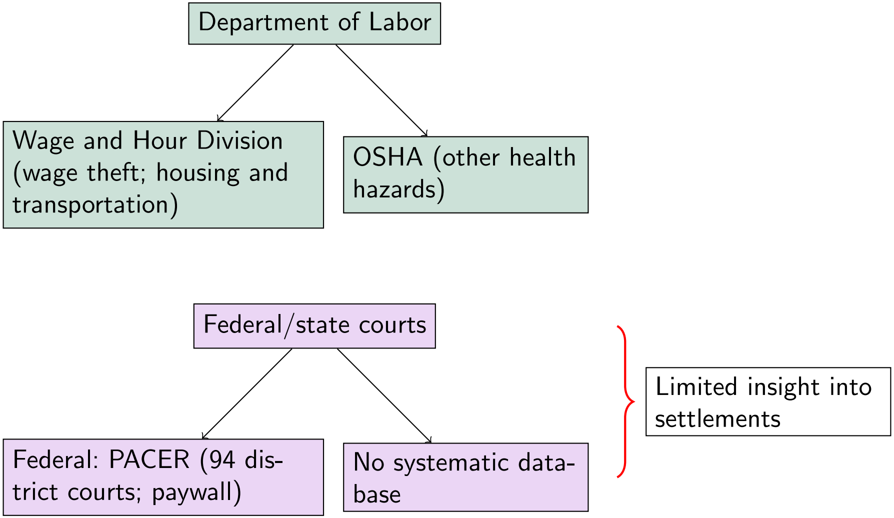
- Some course housekeeping
- Basic command line syntax
- **Recap on SIP partner visit and DataCamp questions**
- Git/GitHub
- LaTeX/Overleaf
- Intro to problem set 1

SIP partner visit

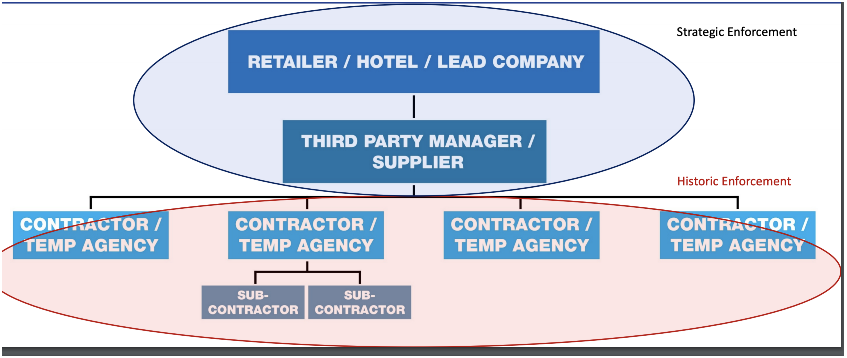
- Our questions/some notes for Daniela and Dave's visit: https://docs.google.com/document/d/1bmmztzKQ2R_ltL-EYkATfBGeYoZJNIwjOHWcBI6E3BY/edit
- A few takeaways on my end:
  1. **Low incentives to report/enforcement fragmentation**
  2. **Challenges of employment structures for legal enforcement**
  3. **What counts as transparency to the public about violations?**
  4. **Burden of proof when identifying repeat violators**
- **Acronyms**: DOL = Department of Labor; WHD = Wage and Hour Division; OSHA = Occupational Safety and Health Administration; H2 = visa name (A and B are subtypes; see SIP page on website)

SIP recap 1: fragmented enforcement landscape

Administrative enforcement; “Private” enforcement


Department of Labor
Wage and Hour Division (wage theft; housing and transportation)
OSHA (other health hazards)
Federal/state courts
Federal: PACER (94 district courts; paywall)
No systematic database
Limited insight into settlements

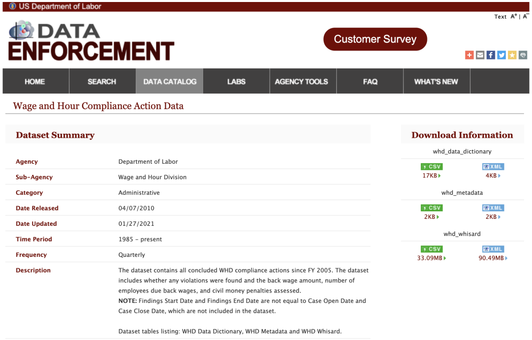
SIP recap 2: which entities does enforcement target?



Source: David Weil, Slides on *The Fissured Workplace*

SIP recap 3: transparency

- Where DOL Wage and Hour Division stores their investigations/violations data— 400,000+ rows in an excel spreadsheet (link)



Wage and Hour Compliance Action Data

Dataset Summary

Agency	Department of Labor
Sub-Agency	Wage and Hour Division
Category	Administrative
Date Released	04/07/2010
Date Updated	01/27/2021
Time Period	1985 – present
Frequency	Quarterly
Description	The dataset contains all concluded WHD compliance actions since FY 2005. The dataset includes whether any violations were found and the back wage amount, number of employees due back wages, and civil money penalties assessed. **NOTE:** Findings Start Date and Findings End Date are not equal to Case Open Date and Case Close Date, which are not included in the dataset. Dataset tables listing: WHD Data Dictionary, WHD Metadata and WHD Whisard.

Download Information

	CSV	XML
whd_data_dictionary	17KB	4KB
whd_metadata	2KB	2KB
whd_whisard	33.09MB	90.49MB

- Benefits and drawbacks in having flags for violations posted alongside a job posting? Other ways to increase transparency?

SIP recap 4: repeat violators

- Example of `MAS LLC` shut down and renamed `MAS LLC 2`
- **Technical question:** what tools would we use to assess similarity in names?
- **Ethical question:** should burden of proof in coding two companies as "likely the same employer" vary based on who's analyzing the data and the stakes (e.g., thesis project; journalist; regulatory agency)?

DataCamp

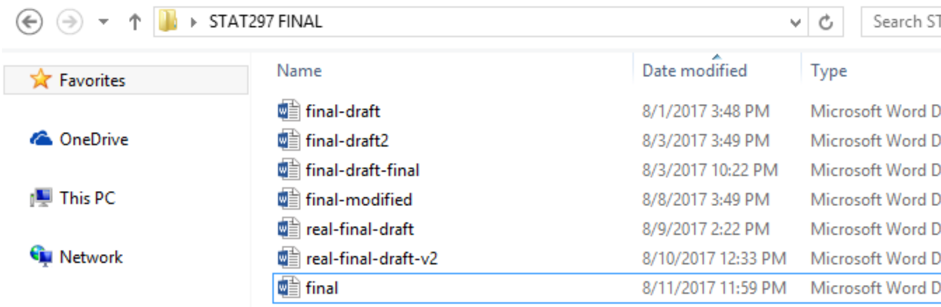
- Questions: https://docs.google.com/document/d/1gYrxrmJcINcNIrs-2tFN8dqLImbvav_VZ6_vA-KiBEM/edit#
- Of the modules, what's most essential for future work in course?
  - **Most important:**
    - Row and column subsetting
    - Aggregation using `groupby` and various functions
    - Iteration/loops
  - **Moderate:** `pivot_tables` (more used is to go from wide to long using `pd.melt`)
  - **Lower:**
    - `set_index` and sort_index methods (as noted in a response to a q, just allows some code shortcuts; can always do same thing with groupby and calling colnames explicitly)
    - matplotlib syntax: for problem sets, you can use any plotting code that works– e.g., matplotlib; ggplot wrapper (plotnine); seaborn. Solutions will be ggplot/plotnine-based

Goal for today's session

- Some course housekeeping
- Basic command line syntax
- Recap on SIP partner visit and DataCamp questions
- **Git/GitHub**[*]
- LaTeX/Overleaf
- Intro to problem set 1

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

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

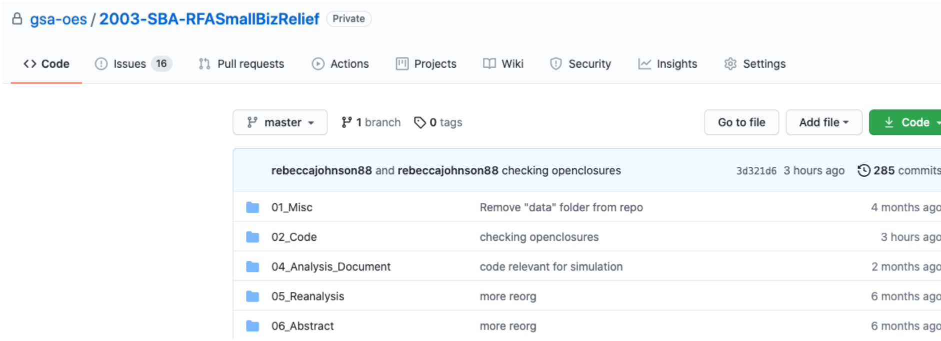
What is Git?

- 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

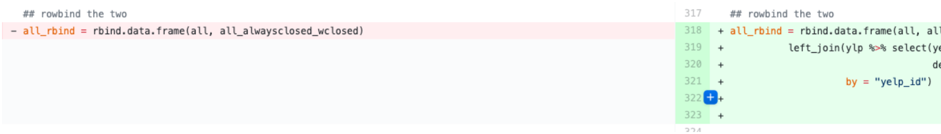

gsa-oes / 2003-SBA-RFASmallBizRelief Private

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, al
+           left_join(ylp %>% select(ye
+                                      de
+                 by = "yelp_id")
+
+
```

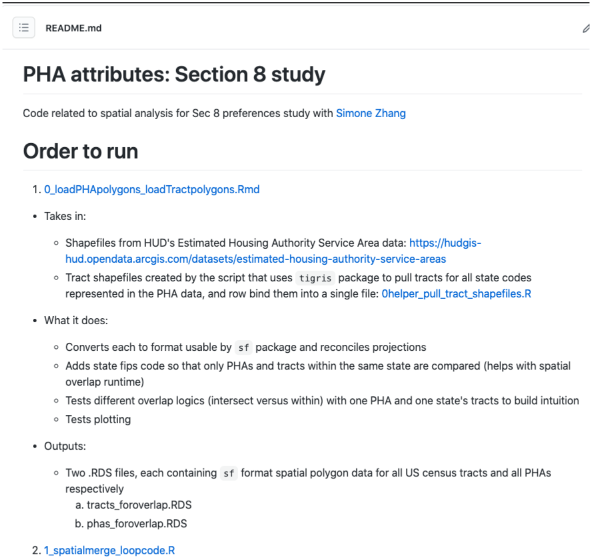
Example repo: public repo

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

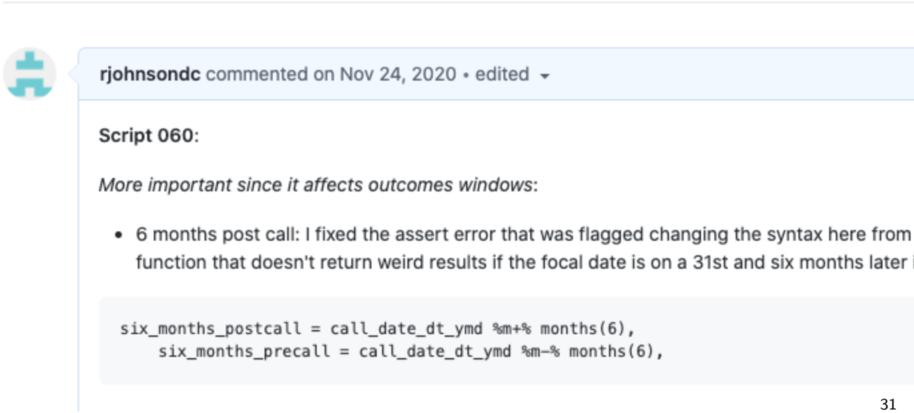
Command line syntax in previous slide is useful for org/reorg. For our class, we'll generally have two directories:

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 -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) 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'll cover in future session

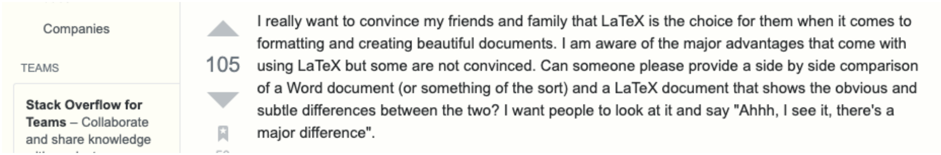
- **Storing your credentials**
- **Tools for more collaborative coding:** branching and pull requests
- **Options to reverse changes**
- **Slightly different cloning structure on jhub**

Goal for today's session

- Some course housekeeping
- Basic command line syntax
- Recap on SIP partner visit and DataCamp questions
- Git/GitHub
- **LaTeX/Overleaf**
- Intro to problem set 1

Overview before activity

- LaTeX: typesetting language
- As discussed in sofware setup, 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



I really want to convince my friends and family that LaTeX is the choice for them when it comes to formatting and creating beautiful documents. I am aware of the major advantages that come with using LaTeX but some are not convinced. Can someone please provide a side by side comparison of a Word document (or something of the sort) and a LaTeX document that shows the obvious and subtle differences between the two? I want people to look at it and say "Ahhh, I see it, there's a major difference".

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

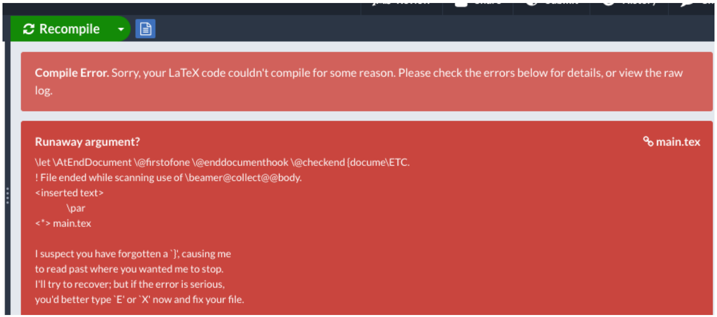
```
\begin{enumerate}
\item First item...
\item
\end{enumerate}
```

- 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: 

Compile Error. Sorry, your LaTeX code couldn't compile for some reason. Please check the errors below for details, or view the raw log.

Runaway argument? main.tex

```
\let \AtEndDocument \@firstofone \@enddocumenthook \@checkend {docume\ETC.
! File ended while scanning use of \beamer@collect@@body.
<inserted text>
          \par
<*> main.tex

I suspect you have forgotten a `}', causing me
to read past where you wanted me to stop.
I'll try to recover; but if the error is serious,
you'd better type `E' or `X' now and fix your file.
```

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
```

Small-group 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`

Goal for today's session

- Some course housekeeping
- Basic command line syntax
- Recap on SIP partner visit and DataCamp questions
- Git/GitHub
- LaTeX/Overleaf
- **Intro to problem set 1**

Substance: racial disparities in felony sentencing

- **Data**: deidentified felony sentencing data from Cook County State's Attorney's Office (SAO)
- Released as part of push towards transparency with election of a new prosecutor in 2016

Opinion

EDITORIAL

Unequal Sentences for Blacks and Whites

By The Editorial Board

Dec. 17, 2016

Earlier this month Kim Foxx, the state's attorney for Cook County, Illinois, which covers Chicago, released six years' worth of raw data regarding felony prosecutions in her office. It was a simple yet profound act of good governance, and one that is all too rare among the nation's elected prosecutors. Foxx asserted that "for too long, the work of the criminal justice system has been largely a mystery. That lack of openness undermines the legitimacy of the criminal justice system." Source

Mechanics

- **When released?** group portion on Wednesday; individual portion on Thursday
- **Where released?** GitHub and jupyter hub
- **When due?** 1159 pm EST on Thursday 04.15
- **How are groups assigned?** Randomly; will post on Slack
- **How will group portion be assessed?** One submission for group portion per group; one grade; yes/no feedback survey after on whether person contributed (defined minimally as showed up for meetings and tried to participate); if consensus is "no," we'll meet and if no extenuating circumstances, I will upweight your individual portion from 20% to 50%
- **Collaboration/resources** I'll create a private slack channel with me + your group; can post code examples freely there for troubleshooting; i'll filter things out to `pythonhelp_general` as relevant in more general form; stack overflow/web-based resources fine but try to cite if using code relatively directly