STAT 214 Spring 2025
Week 1

Zach Rewolinski

Heavily Borrowed from Anthony Ozerov's STAT 215A Materials.

Course Announcements

Questions?

Breakout, ~5min

- We encourage collaboration
  - & will force it later (group projects)
- Introduce yourself to your neighbors
  - Name, program, etc.
  - What did you do over winter break?

Infrastructure for Coding

Scripts vs. Notebooks

Scripts (`.py`)

- Equivalent of `.R`
- Basically just a text file containing code
- Edit in any text editor (vim, emacs, vscode, etc)
- Run using `python my_script.py`. This will run all of the code in the script.

Notebooks (`.ipynb`)

- Equivalent of `.Rmd` or `.Qmd`
- File which can contain code, outputs, and text descriptions
- Edit in a web browser w/ a jupyter server, or vscode with the notebook plugin
- Run by clicking through the cells in the notebook. Cells are individually run, allowing for modular code.

When should you use each?

`.py`

`.ipynb`

When should you use each?

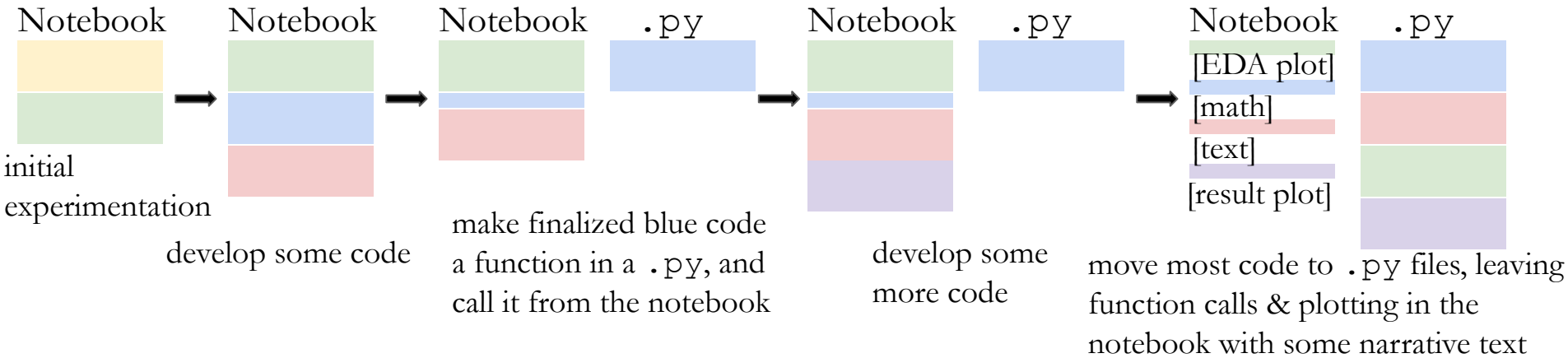
`.py` is good for:

- Functions and classes that you write
- Finalized code
- Version control and collaboration

`.ipynb` is good for:

- Playing around with data
- Developing new code
- Presenting work as a narrative with descriptions of each step

One possible workflow:



Important Tool for Developing Notebooks Alongside .py Files

```
%load_ext autoreload

%autoreload 2
```

Writing Good Code Quickly - Editor Setup

Use a good text editor like vscode (or vim if you enjoy pain) to write your `.py` files.

Helpful features:

- Syntax highlighting: makes it easy to tell by eye what's a string, what's a function, etc.
- Parenthesis matching: it is nice to have some way to tell which close bracket is associated with a particular open bracket
- Plugins: plugins can check Python code style, detect syntax errors, and do other things which make coding easier.

Don't spend too much time trying to optimize your setup from the start.

Python Code Style

Your code should look nice, but we won't dogmatically enforce a code style. A few key points:

- Avoid extremely long lines of code. Keeping below about 120char is good.
- Avoid importing stuff from packages with `from package_name import *`, as this makes it hard to tell what package something came from.
  Instead, use `from package_name import thing_1, thing_2` or simply `import package_name` then use `package_name.thing_1`
- Variables and functions in Python should be written lowercase with underscores to separate words. e.g. `my_variable`. Not following this will expose you as someone new to Python.

Follow PEP8 or the Google Python style guide or something else to keep your code looking clean. If you want, try using a "linter" to enforce a strict style.

Documentation

Your code should be well-documented. This means:

- Comments throughout explaining what is going on.
- Docstrings for functions and classes. Feel free to follow use any style guide for this.

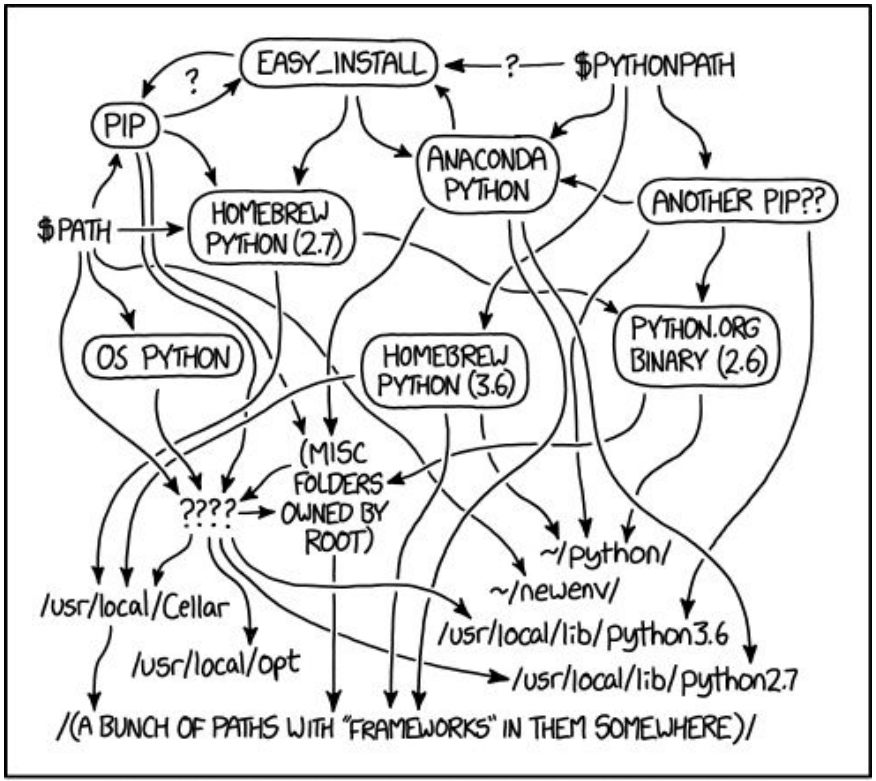
This is something that LLMs are quite good at. The docstring below was written by accepting suggestions from Copilot.

```
# function to invert matrix
def invert_matrix(matrix):
    """
    Invert a matrix.
    Args:
        matrix: (numpy.ndarray) the input matrix to be inverted.
    Returns:
        (numpy.ndarray) the inverted matrix.
    """
    return np.linalg.inv(matrix)
```

Python Environments

Python itself and any packages you install can end up in a number of different places depending on how you install them.

It can get messy.



MY PYTHON ENVIRONMENT HAS BECOME SO DEGRADED THAT MY LAPTOP HAS BEEN DECLARED A SUPERFUND SITE.

Solution: environment management

1. Don't install **`Python`** on its own. Please, just don't.
2. Install **`miniconda`** or **`conda`** (preferably the first). Don't install Anaconda.
3. Create an environment for this class. This environment will contain your Python installation:

   `conda create -n stat214`
4. Activate the environment:

   `conda activate stat214`
5. Install Python in the environment:

   `conda install python`
6. Whenever you do anything in this class (run Python code, run a notebook, install Python packages), do it in a terminal where the environment is activated.
7. To install Python packages, make sure the environment is activated, then use **`pip install`** or **`conda install`**. One package you will need is Jupyter, which can be installed with **`pip install jupyterlab`**.

Using R Too??

Some say that R is better than Python for data visualization (I fall into this camp).

Feel free to use R for this! And for other things if you want.

How to interface R with Python code:

❌ **Use R in Python or Python in R**

R from Python: `rpy2`

Python from R: `reticulate`

- Strange APIs
- Weird datatypes and classes
- Impossible-to-read error messages
- Poor IDE integration
- Will probably break
- Please don't do this

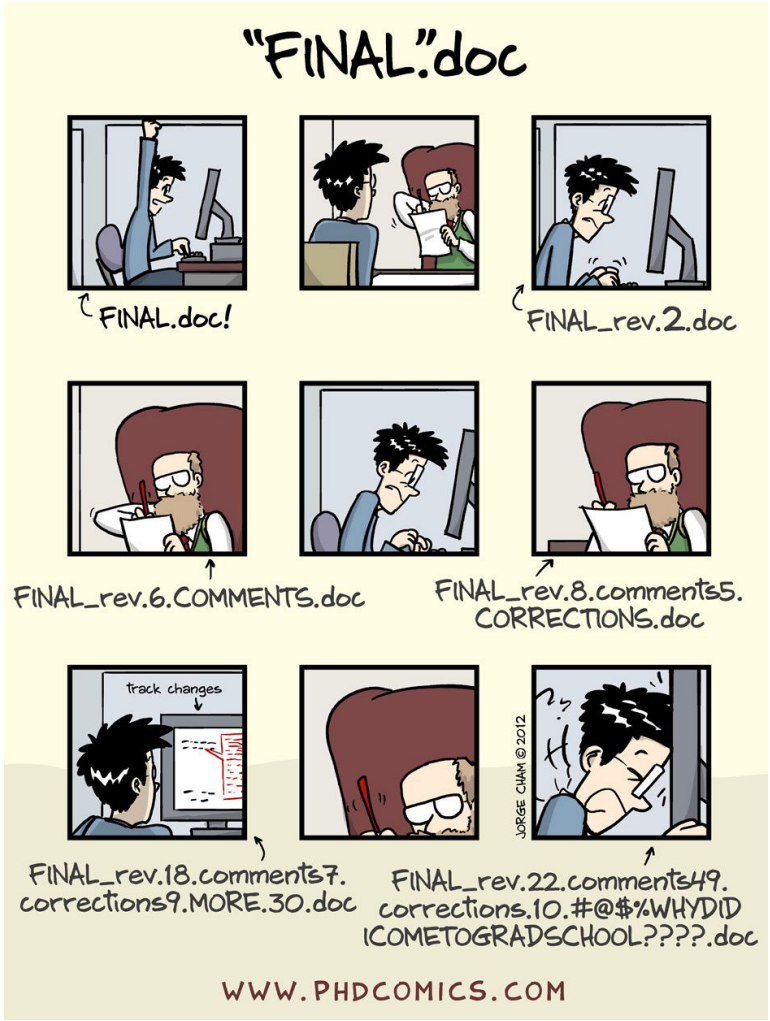
✅ **Use them separately, passing data files between them when needed**

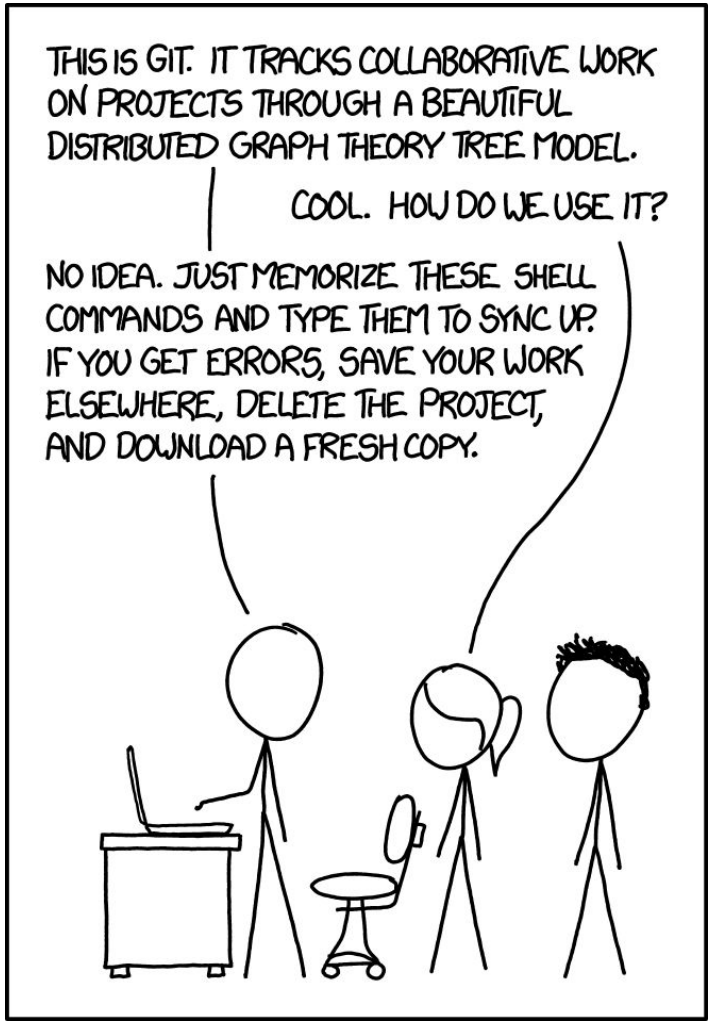
- Write normal Python and R
- Just save your results from Python (.csv maybe), load it in R, and use R like usual

git

What is git?

- A version control system (VCS)
- Stores files as a series of snapshots
- Allows access to all committed steps (i.e. past versions) along the way


"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.#@$%WHYDID
ICOMETOGRADSCHOOL????.doc
JORGE CHAM © 2012
WWW.PHDCOMICS.COM


THIS IS GIT. IT TRACKS COLLABORATIVE WORK ON PROJECTS THROUGH A BEAUTIFUL DISTRIBUTED GRAPH THEORY TREE MODEL.
COOL. HOW DO WE USE IT?
NO IDEA. JUST MEMORIZE THESE SHELL COMMANDS AND TYPE THEM TO SYNC UP. IF YOU GET ERRORS, SAVE YOUR WORK ELSEWHERE, DELETE THE PROJECT, AND DOWNLOAD A FRESH COPY.

git vs. GitHub

git:

- The software which manages git repositories
- Open-source
- Can use locally
- History stored in a .git file in the root directory of your repository

GitHub:

- Company which provides hosting for git repositories.
- Owned by Microsoft
- By far the most commonly used platform for git repositories
- If the repository is public, everyone can see the changes on github.com

Why is git important?

Imagine working on a project with several collaborators…

Using git/GitHub allows everyone to have their own local version of the project while still maintaining a "main" version of the project, hosted remotely on GitHub

You can make changes freely without people seeing what you are doing. You can thoroughly test your changes before adding to the master copy.

Version control!! If your changes create bugs, you can backtrack/revert.

Typical Pipeline

(1) git pull

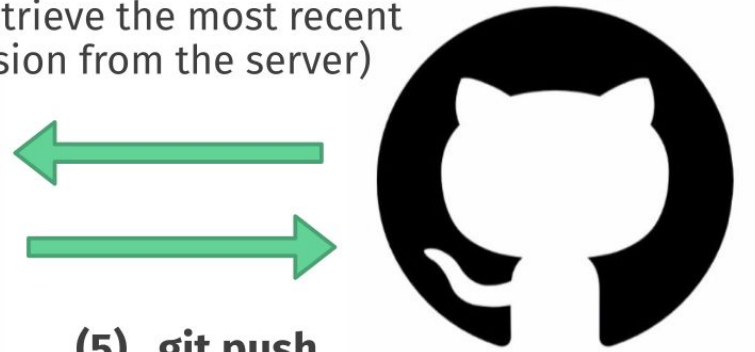
(to retrieve the most recent version from the server)

Remote Repository

(2) make local changes

(e.g. create file called filename.txt)

Local Repo



(5) git push

(make changes available to everyone with access to the repo)

(3) git add filename.txt

(4) git commit -m “add description”

(add and commit when you have made some changes and want to be able to save your current checkpoint as a snapshot)

Warning: Good idea to `git pull` before you `git push` and then you may have to mitigate any merge conflicts.

Two VERY Important GitHub Repositories

`stat-214-gsi`

- For releasing lab assignments & slides
- You only pull from this repo to get materials
- I guess if you want to fix my mistakes you can contribute to it?

Set up your own version using:

`git clone`
https://github.com/zachrewolinski/stat-214-gsi

`stat-214`

- For submitting labs and hw
- You clone the repo, add your finished labs and hw, and push

Follow the instructions at:
https://github.com/zachrewolinski/stat-214-gsi/tree/main/setup
To set it up. Please follow them closely!!!

Let's now set up our repos, look through the lab instructions in disc/week1/lab-instructions.pdf in the GSI repo, and start working on lab0

Don't forget to add me as a collaborator on your stat-214 repo!
(lab0 is due Friday at midnight. Not for a grade, but you must at least submit empty report files so we know your git repo is in working order.)