How To Engineer Big Data

not in the cloud
By Hayden & Alyssa
9am on Tues 2/28/2022

Talk Outline

1. Who are we? (Both)
2. How do you make code that is legible and ready for change? (Hayden)
3. How do you use git/keep track of changes? (Hayden)
4. What does a good coding workflow look like? (Alyssa)
5. How do you make a data pipeline? (Alyssa)
6. What tools are out there to help me deal with data? (Alyssa)
7. Q & A (Both)

Who Are We?

Making Code that is Legible & Ready for Change

Using Git & Keeping Track of Changes

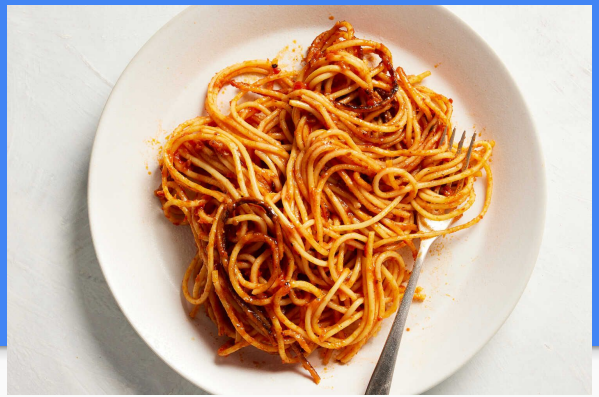
A Good Coding Workflow

What makes a coding workflow good?

(big ol’ disclaimer that this is all my opinion)

- Other people should be able to read & use your end product.
- You should be able to understand & alter your end product if you put it down for a year.
- The end product should work in the way you expect it to (and you should have ways to guarantee or test that)

Don’t Write Spaghetti!



- Spaghetti code contains confusing functions, opaque references, and few comments. It’s tangled! It’s messy! There’s sauce flying everywhere!
- You might be able to understand your spaghetti code, and maybe it’s fine for a one-off testing script, but sometimes you want to write lasagna instead.

Here's Some Spaghetti That I Wrote

```
if data == "wiki":
    prefix = "./article_pickles/"
    content = [
        pickle.load(open(prefix + fname, "rb")) for fname in sorted(os.listdir(prefix))
    ]

    def to_seconds_from_epoch(fname):
        return (
            pd.to_datetime(fname.split("_")[0]) - pd.Timestamp("1970-01-01", tz="UTC")
        ) // pd.Timedelta("1s")

    timestamps = [to_seconds_from_epoch(fname) for fname in sorted(os.listdir(prefix))]

    content = [[w for w in doc if len(w) < 20 and len(w) > 3] for doc in content]
    content = [
        [w for w in doc if w not in wiki_to_remove and wiki_substring_safe(w)]
        for doc in content
    ]
elif data == "tmz":
    pickles_to_load = ["articles_scraped.pkl", "articles_scraped_run_2.pkl"]
    articles = []
    for pkl in pickles_to_load:
        articles.extend(pickle.load(open(pkl, "rb")))
    content = [
        [
            w.strip().lower()
            for w in a[0].split("see also")[0].split("Share on Facebook")[0].split()
```

What makes this spaghetti?
How can I do better?

Make Lasagna

I know this metaphor isn't perfect, but lasagna has layers that are more or less aligned. There's sauce and cheese between them. It's less chaotic. You know what to expect between the layers.

There are good practices and automated tools to help you make lasagna.

Hayden's covered some of them, but I'll run through a few practices & tools for you!

Practices For Making Lasagna

- Plan out your module/the structure of what you're trying to do ahead of time. (yes, actually draw pictures)
- Break up your code into functions and/or classes when possible. Helps with reusability.
- Write docstrings! And comments where they might be needed!
- Keep track of the packages you're using and what versions they are. Also write down any expectations about the data you're using.
- Review each other's code if you're making major changes (this is why pull requests exist!) and run test suites!

Here’s Some OK Code I Wrote

```
import random


def sample_pareto_random_var(x_0, gamma):
    """
    Sample one random draw from a Pareto distribution
    with minimum value x_0 and exponent gamma.
    returns a float.
    """
    p = np.random.uniform()
    x = x_0 * (1 - p) ** (1.0 / (1 - gamma))
    return x


def dc_srgg_polar(colors, C=5.0, little_c=2.0, gamma=2.05, beta=2.5, x_0=1.0):
    """
    Make a degree-corrected soft random geometric graph
    with the following parameters:
    colors: list of colors, one per node
    C: float, controls the maximum value of the radial coordinate of the points in space
    little_c: float, controls the probability of links forming between two edges
    gamma: the degree exponent for the Pareto distribution. >2.
    beta: the degree exponent for the connection probability. >2.
    x_0: the minimum value of a node's degree

    Returns adj, a dictionary of adjacency indicators,
    and coords, the polar coordinates of the nodes.
    """
    X = np.random.uniform(size=len(colors))
```

What did I do right?
What could be better?

Tools For Making Lasagna

- pip: pip is a package manager. You can get a list of the packages in your environment with the command `pip freeze`, and make a file of requirements for others to use with `pip freeze > requirements.txt`.
- venv: virtual environments allow you to have separate dependencies for each project. Make one for each project: `virtualenv project_name`
- Linters: linters make sure that your code doesn't have any obvious bugs and style errors. If you're worried about adding bugs, a linter can add some peace of mind.
- Auto-formatters: format your code so that (e.g.) multi-line function calls are readable and not ugly. A good one for Python is called `black`.

More Tools For Making Lasagna

- pytest: standard testing library for Python (iirc). Has a good deal of functionality and is user-friendly.
- mock: lets you make fake versions of APIs or other functions to isolate modules you need to test.
- github actions: you can set these up to run a linter and/or a test suite when someone pushes code to your repository. You can also make a git hook, but that's a bit more involved.

A Process I Try To Follow

1. Figure out what I’m doing; draw a picture and set up skeleton functionality.
2. Fill in the functionality, testing as I go. Establish input and output ee

Data Pipelines

Tools For Dealing With Data