BIOSTAT 821

Objective

Build data science solutions that

- work
- work next year
- work with different input
- work on someone else's computer
- can be maintained/updated

... without undue difficulty

Course format

in-class: ~50% content (lecturing, examples), ~50% active (group exercises)

out-of-class: ~1-2 hours content (reading, watching), ~5 hours "development"

Development activities

- writing code
- writing tests
- writing documentation
- fixing/improving any of the above
- reviewing any of the above
- miscellanea
    - e.g. struggling with git

"Development" includes all of the above!

Basic tools

- git
- Python
- VS Code

Portability example

Alice writes a Python tool and pushes it to GitHub. She is a friendly developer and also documents what version of Python and numpy she used.

Bob downloads Alice's tool. He has different Python and library versions on his machine that he is using for other projects.

How can Bob run Alice's code? Multiple "environments".

Environments

- What version of libraries (numpy, etc.)?
- What version of Python (2.7.x, 3.10.x, etc.)?
- What operating system/architecture?
    - We'll get to this later, with Docker.

Tools:

- `pyenv` handles multiple Python versions.
- `venv` handles multiple library versions.
- `conda` handles multiple Python and library versions.

pyenv (UNIX)

pyenv is used to install Python and to switch between Python versions.

- Specific versions can be assigned to directories (projects). This configuration is then inherited by subdirectories.

pyenv (UNIX) setup

First, install pyenv: https://github.com/pyenv/pyenv#installation

Confirm that this worked with `which pyenv`.

Make sure to set up your shell environment for pyenv.

```
pyenv install 3.11.1 # install a specific version of Python
mkdir biostat821 && cd biostat821 # create project folder
pyenv local 3.11.1 # choose Python version for project
```

Confirm that this worked with `which python`. The path should include "pyenv". `python --version` should match the version that you installed.

pyenv (Windows)

pyenv-win

-OR-

use Windows Subsystem for Linux

venv

venv comes standard with Python and is independent of the Python version, mostly.

```
python -m venv venv # create environment
source venv/bin/activate # activate environment
```

The terminal will show "(venv)" indicating that the environment is active.

Conda

Conda can be used to manage both Python and library versions.

- Conda and pyenv can coexist (at least on Mac/Linux).
  - To prevent activating the base conda environment by default: `conda config --set auto_activate_base false`
- `pip` and Conda don't always play nicely together.
  - To avoid most issues, only install packages using `pip install`.

Fun fact: conda can be installed by pyenv 🤯

VS Code

https://code.visualstudio.com/

VS Code configuration

```
pip install mypy pycodestyle pydocstyle black
```

Make sure that the Python extension is installed in the Extensions tab of the left toolbar.

In Settings,

- search "python lint enable" and enable:
  - mypy
  - pycodestyle
  - pydocstyle
- search "format on save" and enable it
- search "python formatting provider" and choose "black"

Activity: hello world

Write a file "hello_world.py" such that `python hello_world.py` prints "Hello world!"

The linters may complain. We'll deal with them later.

Resources

- https://lynn-kwong.medium.com/how-to-create-virtual-environments-with-venv-and-conda-in-python-31814c0a8ec2