Lecture Week 6

Software Engineering and Data Science

Portfolio 2 Release:

Data driven prediction models of energy use of appliances in alow-energy house

Software Engineering and Data Science

- Are notebooks a good idea?
- If not, what is the alternative?
- How can we write re-usable modules?
- Managing dependencies
- Testing in Data Science

Notebooks Considered Harmful

- I don't Like Notebooks (https://www.youtube.com/watch?v=7jiPeIFXb6U) by Joel Grus
  - Out of order execution
  - Discourage modularity and testing
  - Don't provide help writing code
  - Hinder reproducible and extensible Science
- Some of these are interesting because they are suppposed to be advantages of notebooks

So why do we use them?

- If notebooks are so bad, why do we (teach you to) use them?
- Web based interface is easy to work with
- Interactive nature of notebooks easy to experiment
- See the results of code immediately
- Present analysis mixing code, text, data and graphs
- Me: my slides can contain running code examples

Can we keep these advantages and avoid the harmful featues?

Out of order execution

- Notebooks are interactive and based on cells
- Run a cell once or many times
- Order of execution is not defined by the document
- Unlike almost any other code you will write
- A cause of confusion and error

Discourage Modularity and Testing

- One Notebook == One Analysis
- Big blocks of code, defining functions, all in one notebook
- Spread through the notebook
  - References to data files
  - Configuration
  - Dependencies (import statements)
- How do we test code written in a notebook?

No help when writing code

- Notebooks is a very poor programming environment by modern standards
- Code Editors: VS Code, Sublime, Atom, Eclipse, IntelliJ Idea
- Understand langauges and libraries
- Provide context sensitive help
- Help you check syntax, style of code
- Make running tests easy

Hinder Reproducible and Extensible Science

- **This is big** - thse are some of the reasons we use notebooks
- Notebooks make it easy to run the exact same experiment
  - if you can install the same software
  - and you have the data
  - and the file paths work
  - and they din't rely on out-of-order execution
- Can we extend or adapt?
  - yes, by editing the notebook

What's the answer?

- Some of these concerns are about habit/practice
- If we are careful, we can do the right thing
- Other things are just flawas in the Jupyter eco-system
- Be aware of the concerns
- Broaden your knowledge of alternate tools and methods

Python Outside of Notebooks

- We can write Python code outside of notebook files
- Write xyz.py file and run them
  - Code editors for writing Python
  - How do you run Python files?
- As the code grows, modularise in multiple Python files
- Thing about packaging commonly used code

Project Structure

- How to structure a Python project directory
- Different structures for different tasks
  - Standalone project
  - Reusable Python module
- The Python CookieCutter (https://cookiecutter.readthedocs.io/en/1.7.2/) utility is useful
  - Initialises a project directory from a template
  - People publish best-practices templates
    - Data Science Project (http://drivendata.github.io/cookiecutter-data-science/)
    - Python Package (https://github.com/kragniz/cookiecutter-pypackage-minimal)
    - Creating Custom Project Templates (https://towardsdatascience.com/cookiecutter-creating-custom-reusable-project-templates-fc85c8627b07)
  - Demo...

Using Makefiles (Advanced)

- The Data Science Project CookieCutter template uses Make (https://www.gnu.org/software/make/)
- Make is a tool for managing tasks and dependencies
- Originally for compiling software, can be used for any tasks
- Useful for Data Science Projects (http://zmjones.com/make/)

Command Line Scripts

- A Python script intended to be run from the command line
- How do you run them?
  - Need a command line prompt (MacOS: Terminal, Windows: Command Prompt)
  - But could be within your environment (Jupyter, VS Code)
  - Must have your Python environment loaded
  - Then python xyz.py arg1, arg2, arg3
- (or from a Makefile)

Why Command Line scripts

- Like a notebook
  - easy to re-run your process
  - can edit-run-edit-run to debug
- Unlike a notebook
  - no out-of-order execution
  - encourages parameterisation
  - output to terminal or saved to a file
  - more easily testable

Experimental Configuration

- Most experiments we do have some configuration parameters
  - Where is the input data
  - What variables to use to train the model
  - Number of clusters to look for in KMeans clustering algorithm
- For more complex machine learning models there are more parameters
- Easy just to include these in code

Experimental Configuration

- Alternative is to include all configuration settings in a config file
- Read settings in your script/notebook
- No (or fewer) hard-coded values
- Means that changing settings is easier
- Encourages you to think about portability and modularity

Alternative formats are YAML, JSON, INI - see this article (https://martin-thoma.com/configuration-files-in-python/) for a summary

Testing in Data Science

- Tests give you confidence that your code does what you expect
- Help (force) you to think about what you expect
- Make your assumptions explicit and checkable
- Help to ensure that changes don't break assumptions

Examples:

- TDD in a Data Science Workflow (https://towardsdatascience.com/tdd-datascience-689c98492fcc)
- Data Testing Tutorial (https://ericmjl.github.io/data-testing-tutorial/)
- Getting Started Testing: pytest edition (https://nedbatchelder.com/text/test3.html)