Python pipelines for data analysis (using Dask!)

Joe Donovan

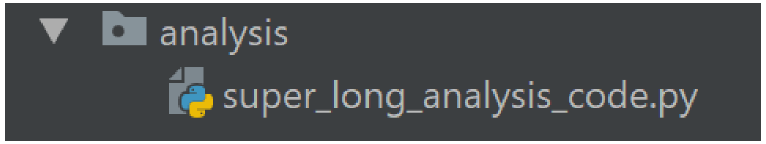
Pop quiz - data and figure naming

Which is better?

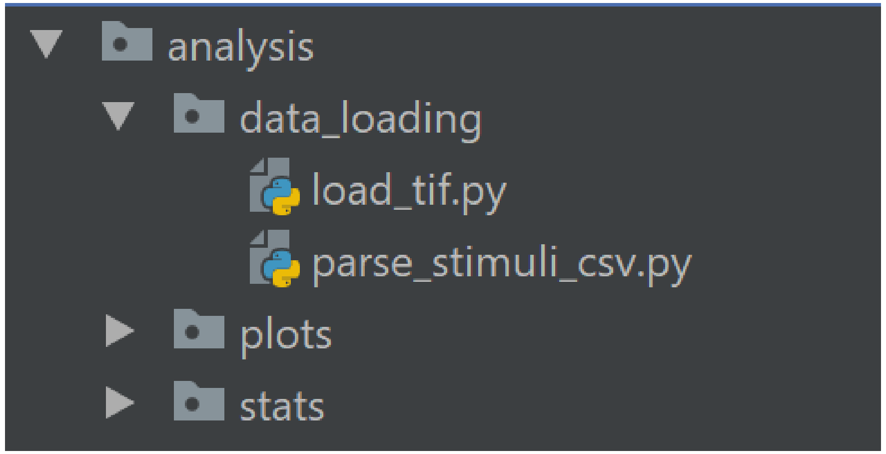
"data1.csv"		"2018_12_11_experiment_5.csv"
"plot1.png"	or	"exp5 timeseries mean.png"

Pop quiz – code organization

Which is better?



or



Pop quiz – don't repeat yourself

Which is better?

```
dataset1 = load_data('datapath1')
dataset1 = some_function(dataset1)
dataset1_temp = dataset1 * 1000
dataset1mean = dataset1_temp.mean(axis=1)

dataset2 = load_data('datapath2')
dataset2 = some_function(dataset2)
dataset2_temp = dataset2 * 1000
dataset2mean = dataset2_temp.mean(axis=1)

dataset3 = load_data('datapath3')
dataset3 = some_function(dataset1)
dataset3_temp = dataset1 * 1000
dataset3mean = dataset3_temp.mean(axis=1)
```

or

```
def process_data_means(datapaths):
    """ Returns means from processed data"""
    datasets = [1000 * some_function(load_data(dp)) for dp in datapaths]
    return [ds.mean(axis=1) for ds in datasets]


means = process_data_means(['datapath1', 'datapath2', 'datapath3'])
```

What do we want for data analysis process?

- Interactive/exploratory
  - Use Jupyter notebooks
- Reusable (across datasets and workflows)
  - Move solidified processing steps into **reusable functions**
  - For similar datasets, import and use common code as much as possible

What do we want for data analysis process?

- Consistent and reliable
  - Data naming consistent – each new dataset is a new filename
  - **Version control code**
  - Fix random seeds and record library versions

What do we want for data analysis process?

- Robust and correct
  - Use assert statements
  - Test, simulate, and shuffle

What do we want for data analysis process?

- Efficient
  - Auto (re)run code as it (or the data) change
  - Not currently easy
- Can cache intermediate results
  - But out of date cached results are really bad
- Dask has “experimental” Opportunistic Caching

Make your analysis pipeline easy to rerun

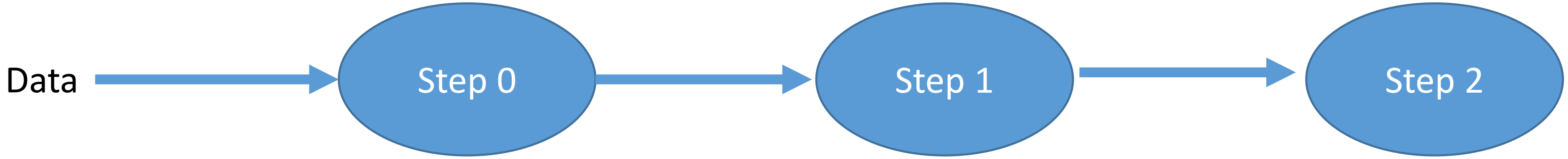
- Organize the data going in
- Plan out the figures and stats coming out
- Can save matplotlib figure panels in vector format (.pdf)
- Import as a linked file into illustrator

- Define common parameters (ie plot colors) in one place

- Use functions!
  - Several levels of functions is fine

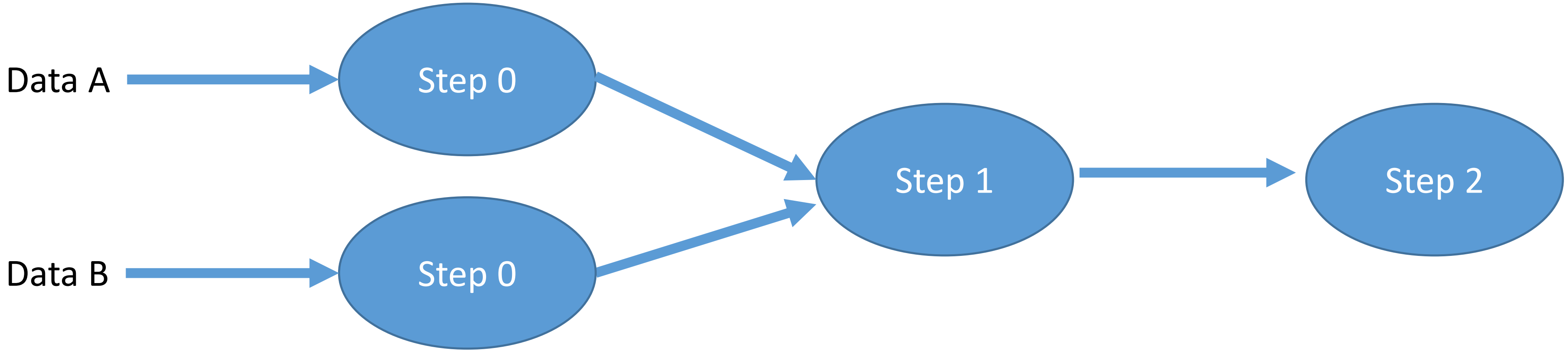
Split your analysis into steps

- Reusable
- Each step (task) requires some inputs, and produces some outputs



- Is a good mental model a "chain of tasks"?
  - Almost
  - What about loading/combining over multiple datasets?

Task graph



- Graph (connected nodes)
  - Each node is a processing step/task
- Directed
  - Receives some input, and produces different outputs
- Acyclic
  - You can’t require A to compute B, while also requiring B to compute A

Task graph and data pipeline libraries

- Can make a data pipeline in pure python

- Luigi
- pydoit
- Airflow
- Datajoint
- **Dask**
- And many more

Why dask?

- Executes task graphs
- Friendly and familiar interface
- Distributed - scales across cores and even to cluster-level
- On a single machine, changes data size limit from memory to HD
- Visualization and debugging

Dask organization

- Low-level: task graphs and task schedulers
  - Executes your task graph
- High-level: replacements for np array, pandas dataframe, lists (dask bag)
  - Act similar, but easily parallelized/distributed

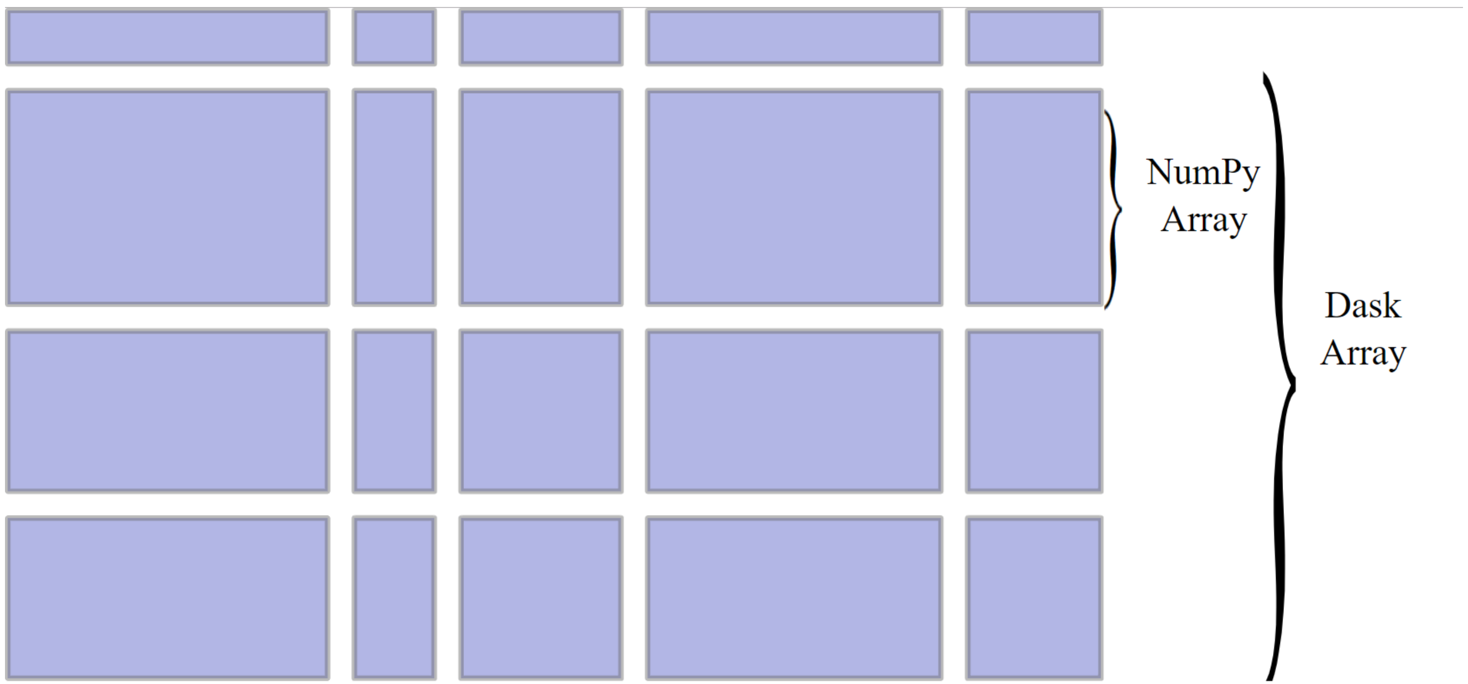
Creating ‘lazy’ task graphs - Dask delayed

- How to define a task graph, without executing it when we define it?

- Be lazy!

- dask.delayed – wraps a task, and returns a promise now of work to be done later

Dask array



- Supports subset of np array methods

Dask performance tips

- http://docs.dask.org/en/latest/delayed-best-practices.html

- Functions shouldn’t change their inputs
- Break computation into many ‘reasonably hard’ chunks
- Minimize passing around huge arrays

Distributing work

- Not every problem distributes well
- Multiple data items are often a good place for parallelization

- Dask comes with some ‘clever’ functions built in (ie filtering)
- Dask ML

- How to keep the environments consistent?
    - Kubernetes