Hamilton Global User Group March 2024 Meetup

What is Hamilton?
Hamilton helps data scientists and engineers define testable, modular, self-documenting dataflows, that encode lineage and metadata. Runs and scales everywhere python does.

Icebreaker: Name and what you’re using Hamilton for/looking for.

DAGWORKS March 2024

Agenda

1. Community Spotlight
2. Deep Dive
3. Open 🎤

Community Spotlight:
🔎 "**Hamilton as a Feature Catalog**"

by Roel Bertens.

Deep Dive: Reuse/Parameterization of Hamilton Code

DAGWORKS

Reuse / Parameterization of Hamilton code

Motivations:

- I want to reuse my prior work
  - E.g. data set cleaning - I want to apply the same transformations on another dataset
  - E.g. I have a model pipeline - I want to create many models with it
- I would like to make my code DRYer
  - E.g. functions do the same thing but with different inputs
- I cannot have two "functions" named the same in a single DAG
  - Hamilton enforces 1:1 output → function.
    - Cannot have two functions named "mean" in the same DAG.

Step 1: Understanding what you really want:

Things to get clear on:

1. *configuration* vs *input* vs *output*?
2. Do I need everything in a single DAG?
3. How often is change going to occur?
   a. What friction do I want for change?

Step 1: Understanding what you really want:

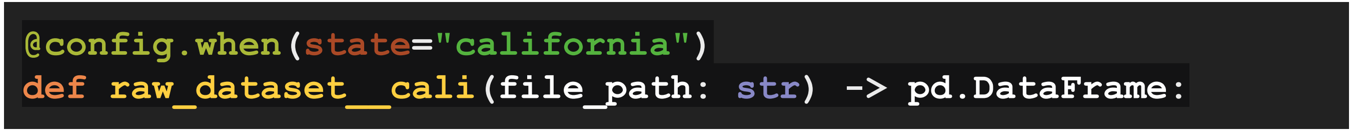
Configuration vs Input vs Output

Configuration shapes the DAG.

Inputs are related to the values processed.

Outputs are what you request to be computed (passed in to .execute() or .materialize())

Config →



```
@config.when(state="california")
def raw_dataset__cali(file_path: str) -> pd.DataFrame:
```

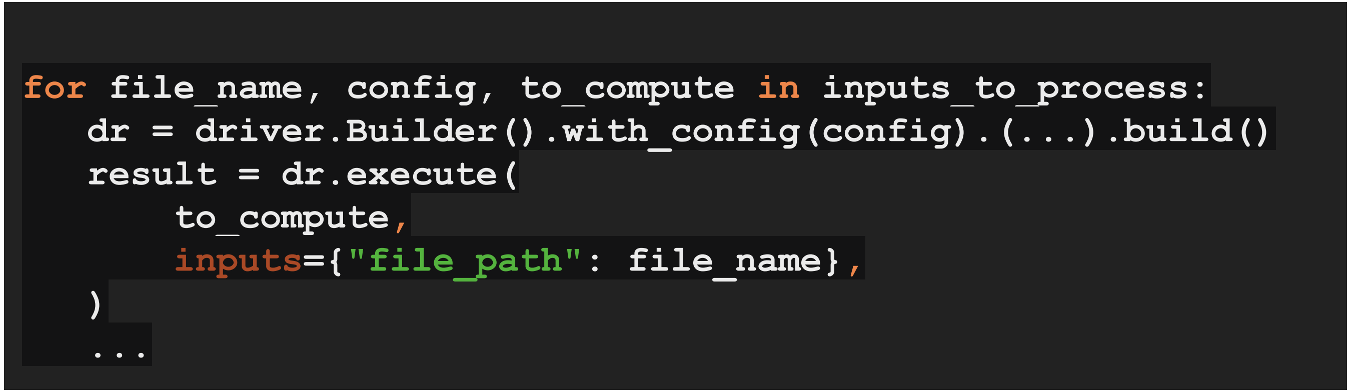
^--- **Input**

Outcome: the parameters of reuse you need.

Step 1: Understanding what you really want:

Do I need everything in a single DAG?

KISS:



```
for file_name, config, to_compute in inputs_to_process:
   dr = driver.Builder().with_config(config).(...).build()
   result = dr.execute(
       to_compute,
       inputs={"file_path": file_name},
   )
   ...
```

Q: Will a single DAG make it easier to operate and/or understand?

Step 1: Understanding what you really want:

How often is change going to occur?

1. Will it occur often?
   a. **No**: is this a good use of your time?
   b. **Yes**: where do you want the friction? (next question)
   c. Maybe: ?
2. What do you want people to change / update (PR process or not?)?
   a. Function code
   b. Configuration – note: configuration is often treated like code!
   c. Inputs
   d. Driver script code

Step 1: Understanding what you really want:

Things to get clear on:

1. *configuration* vs *input* vs *output*?
2. Do I need everything in a single DAG?
3. How often is change going to occur?
    a. What friction do I want for change?

End Outcome:

- Knowing the dimensions of reuse.
- Knowing the importance/value you're optimizing for.

Step 2: Implementation Options

To start: get something working for one case.

Then it depends...

- For-loops over multiple drivers / Hamilton within Hamilton
- Making names all unique
- Use @subdag to create a large DAG (did overview last month)
- Custom result builders
- Parallelizable + Collect
- @resolve [+ @inject (or other decorators)]
- @pipe
- A combination of the above

Step 2: Implementation Options

Code walkthrough:

https://github.com/DAGWorks-Inc/hamilton-tutorials/blob/main/2024-03-19/march-meetup.ipynb

🗓️ Next month - April 16th:

Want to speak?

🎤 Open Mic.

📝 Survey - https://forms.gle/mjCtbCNWszpgFxuj7

FIN. Thanks for coming!

DAGWORKS