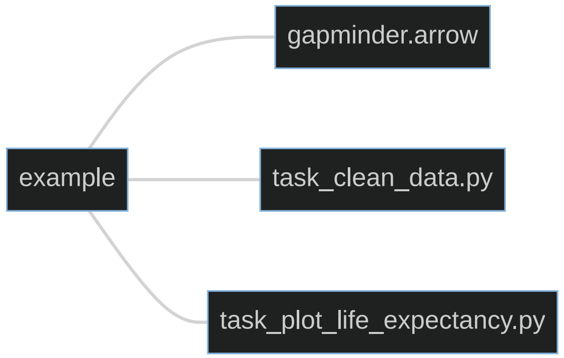
Effective Programming Practices for Economists

Reproducible Research

What does pytask do?

Janoś Gabler and Hans-Martin von Gaudecker

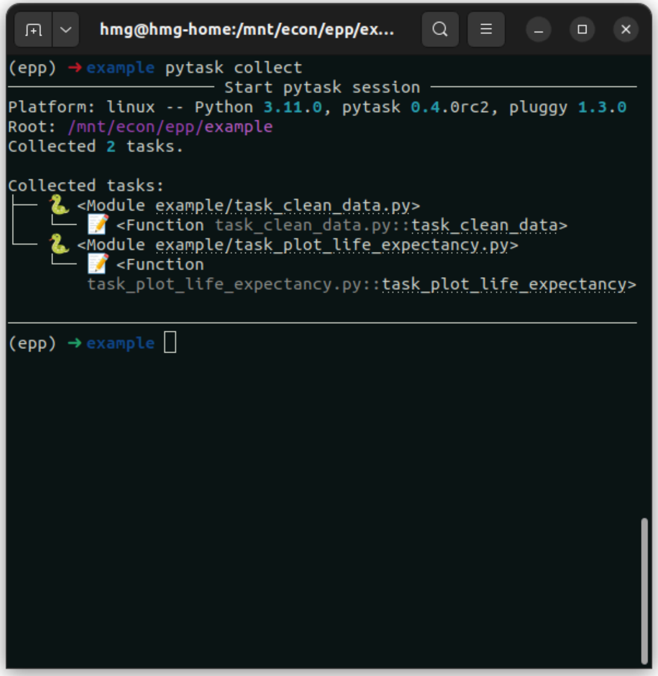
A tiny example project



- `example/task_clean_data.py`
  - Contains the function `task_clean_data`
  - If called, the function reads in `example/gapminder.arrow` and produces `example/bld/data.pkl`
- `example/task_plot_life_expectancy.py`
  - Contains the function `task_plot_life_expectancy`
  - If called, the function reads in `example/bld/data.pkl` and produces `example/bld/life_expectancy.svg`

Step 1: collection

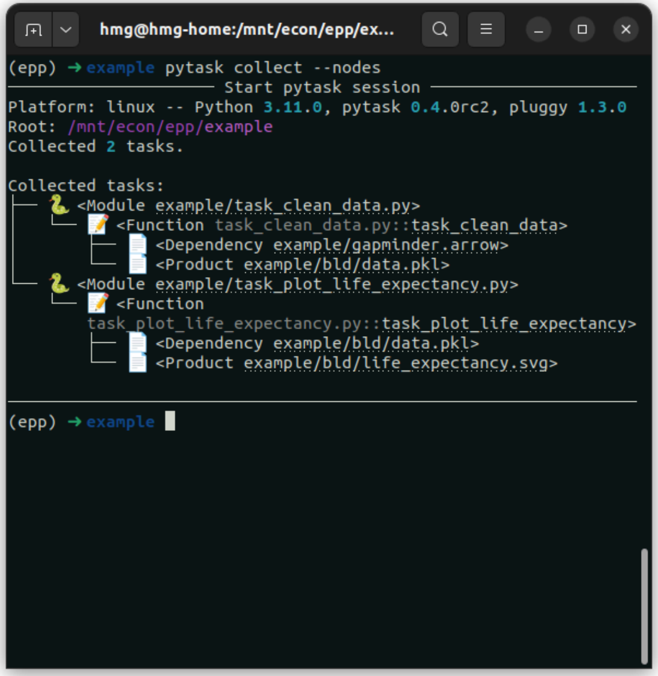
- Go through all folders in working directory
- Collect all files with name `task_XXX.py`
- Go through those files and collect all functions that start with `task_`
- Task functions and their (default) inputs will be used to construct the workflow



```
(epp) → example pytask collect
──────────────────────── Start pytask session ────────────────────────
Platform: linux -- Python 3.11.0, pytask 0.4.0rc2, pluggy 1.3.0
Root: /mnt/econ/epp/example
Collected 2 tasks.

Collected tasks:
├── 🐍 <Module example/task_clean_data.py>
│   └── 📝 <Function task_clean_data.py::task_clean_data>
└── 🐍 <Module example/task_plot_life_expectancy.py>
    └── 📝 <Function
        task_plot_life_expectancy.py::task_plot_life_expectancy>

──────────────────────────────────────────────────────────────────────
(epp) → example
```

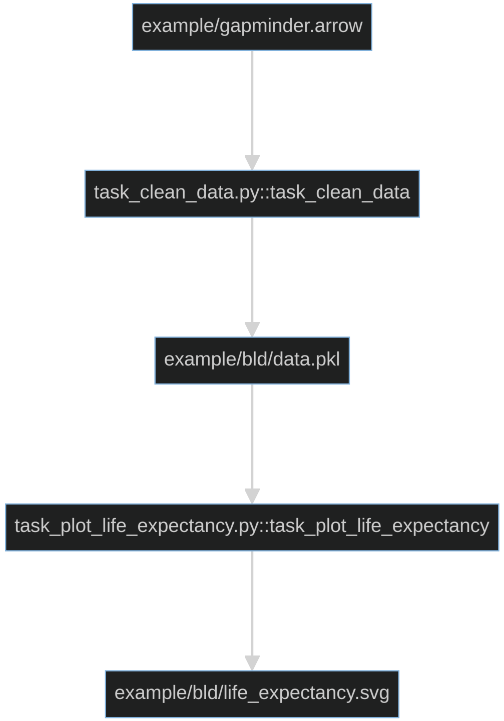
Step 2: Dependency graph (DAG)

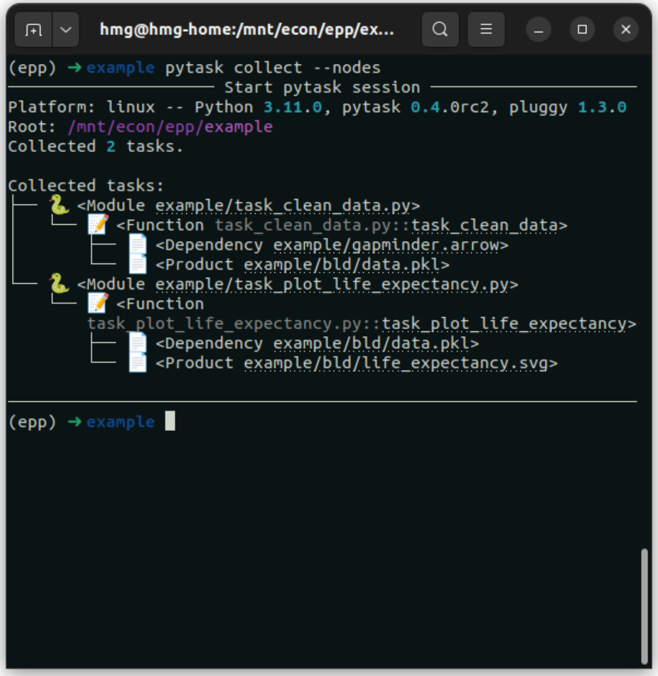
- Inspect function signatures to build a dependency graph
- `produces` describes function output
- Other arguments are function dependencies
- DAG structure enables to determine an order of execution that respects dependency structure (topological sort)



```
(epp) → example pytask collect --nodes
──────────────────────── Start pytask session ────────────────────────
Platform: linux -- Python 3.11.0, pytask 0.4.0rc2, pluggy 1.3.0
Root: /mnt/econ/epp/example
Collected 2 tasks.

Collected tasks:
├── 🐍 <Module example/task_clean_data.py>
│   └── 📝 <Function task_clean_data.py::task_clean_data>
│       ├── 📄 <Dependency example/gapminder.arrow>
│       └── 📄 <Product example/bld/data.pkl>
└── 🐍 <Module example/task_plot_life_expectancy.py>
    └── 📝 <Function
        task_plot_life_expectancy.py::task_plot_life_expectancy>
        ├── 📄 <Dependency example/bld/data.pkl>
        └── 📄 <Product example/bld/life_expectancy.svg>

──────────────────────────────────────────────────────────────────────

(epp) → example █
```

Can you see the DAG?


example/gapminder.arrow
task_clean_data.py::task_clean_data
example/bld/data.pkl
task_plot_life_expectancy.py::task_plot_life_expectancy
example/bld/life_expectancy.svg



```
(epp) ➜ example pytask collect --nodes
──────────────────────── Start pytask session ────────────────────────
Platform: linux -- Python 3.11.0, pytask 0.4.0rc2, pluggy 1.3.0
Root: /mnt/econ/epp/example
Collected 2 tasks.

Collected tasks:
├── 🐍 <Module example/task_clean_data.py>
│   └── 📝 <Function task_clean_data.py::task_clean_data>
│       ├── 📄 <Dependency example/gapminder.arrow>
│       └── 📄 <Product example/bld/data.pkl>
└── 🐍 <Module example/task_plot_life_expectancy.py>
    └── 📝 <Function
        task_plot_life_expectancy.py::task_plot_life_expectancy>
        ├── 📄 <Dependency example/bld/data.pkl>
        └── 📄 <Product example/bld/life_expectancy.svg>

──────────────────────────────────────────────────────────────────────
(epp) ➜ example
```

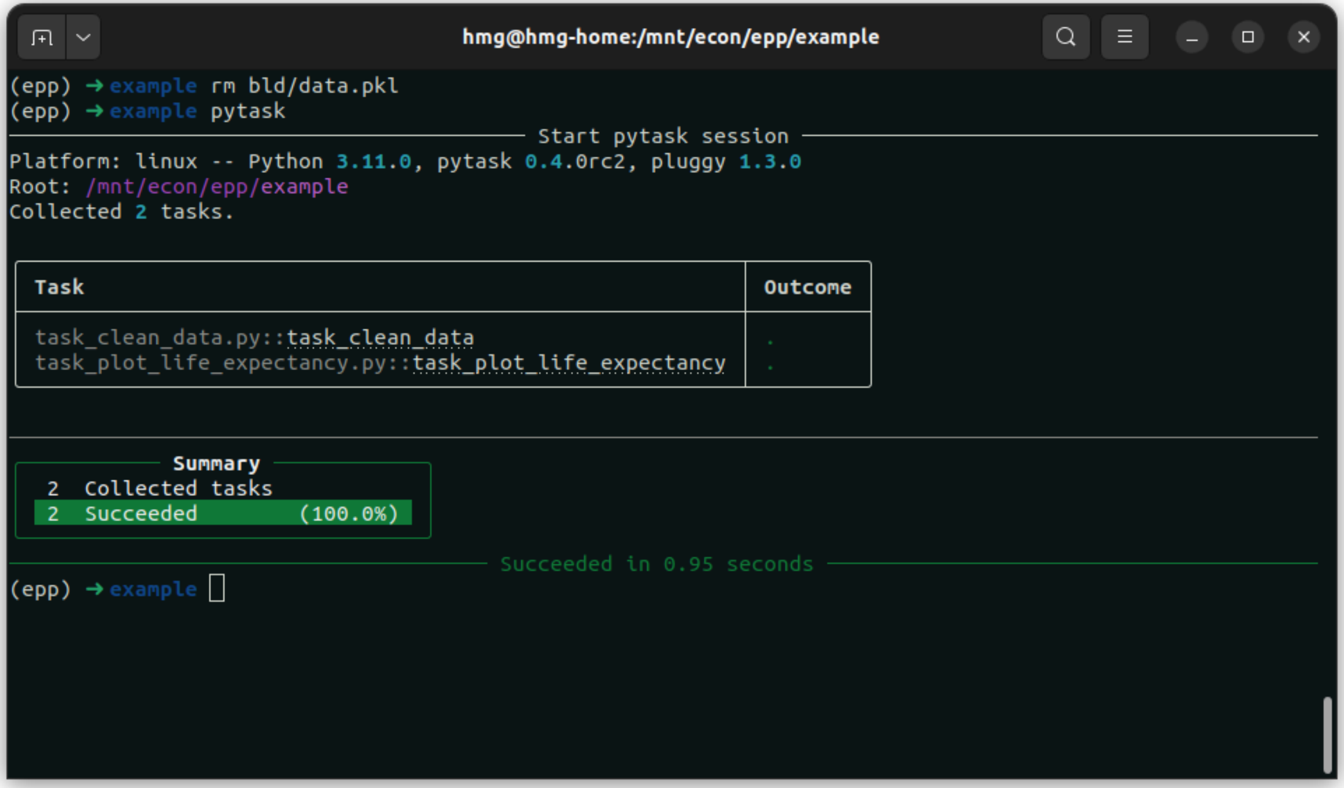
Step 3: Track changes and execute

- Pytask knows which files should need to be generated
- Also keeps track on when code or products have changed
- Functions are only run if:
  - They have changed
  - A dependency has changed
- Huge time savings in large empirical projects!

Run for the first time

Delete plot and run again

Delete cleaned data and run again



```
(epp) → example rm bld/data.pkl
(epp) → example pytask
──────────────────────── Start pytask session ────────────────────────
Platform: linux -- Python 3.11.0, pytask 0.4.0rc2, pluggy 1.3.0
Root: /mnt/econ/epp/example
Collected 2 tasks.

┌──────────────────────────────────────────────────────────┬─────────┐
│ Task                                                     │ Outcome │
├──────────────────────────────────────────────────────────┼─────────┤
│ task_clean_data.py::task_clean_data                      │ .       │
│ task_plot_life_expectancy.py::task_plot_life_expectancy  │ .       │
└──────────────────────────────────────────────────────────┴─────────┘

──────────────────────────────────────────────────────────────────────
┌──────────── Summary ────────────┐
│  2  Collected tasks             │
│  2  Succeeded          (100.0%) │
└─────────────────────────────────┘
────────────────────── Succeeded in 0.95 seconds ──────────────────────
(epp) → example
```