

Effective Programming Practices for Economists

Reproducible Research

Re-using pytask functions

Janoś Gabler and Hans-Martin von Gaudecker

1 tasks, 2 products



2 tasks, 1 product each



Contents of task_plot_life_expectancy.py

```
BLD = Path(__file__).parent / "bld"

products = {
    "Asia": BLD / "life_expectancy_asia.svg",
    "Europe": BLD / "life_expectancy_europe.svg"
}

def task_plot_life_expectancy(
    data_file=BLD / "data.pkl",
    produces=products,
):
    df = pd.read_pickle(data_file)
    for region, fig_file in produces.items():
        fig = _plot_life_expectancy(df[df["continent"] == region])
        fig.write_image(fig_file)
```

Contents of task_plot_life_expectancy.py

```
from pytask import task

BLD = Path(__file__).parent / "bld"

for region in ("Asia", "Europe"):

    @task(id=region)
    def task_plot_life_expectancy(
        data_file=BLD / "data.pkl",
        produces=BLD / f"life_expectancy_{region.lower()}.svg",
        region=region,
    ):
        df = pd.read_pickle(data_file)
        fig = _plot_life_expectancy(df[df["continent"] == region])
        fig.write_image(produces)
```

Verify Dependency graph (DAG, tree)

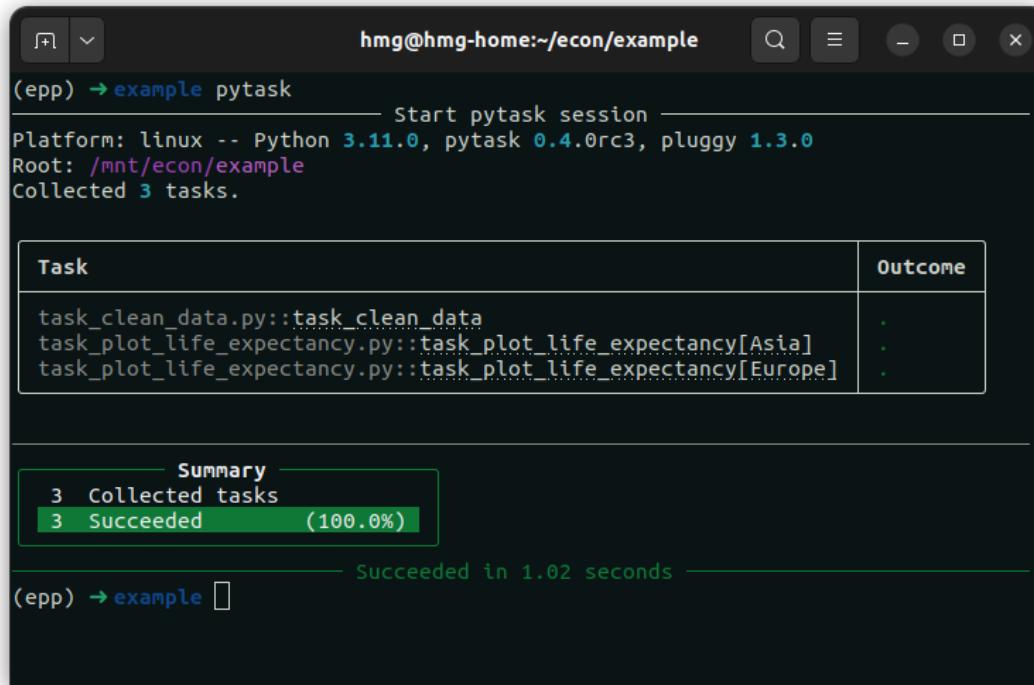
- Inspect function signatures to build a dependency graph
- Tasks for each region appear
- Additional dependency: `region` variable (ignore for now)

```
hmg@hmg-home:~/econ/example
(epp) → example pytask collect --nodes
Start pytask session
Platform: linux -- Python 3.11.0, pytask 0.4.0rc3, pluggy 1.3.0
Root: /mnt/econ/example
Collected 3 tasks.

Collected tasks:
└─ <Module example/task_clean_data.py>
    └─ <Function task_clean_data>
        └─ <Dependency example/gapminder.arrow>
            └─ <Product example/bld/data.pkl>
└─ <Module example/task_plot_life_expectancy.py>
    └─ <Function task_plot_life_expectancy>
        └─ <Dependency example/bld/data.pkl>
        └─ <Dependency example/task_plot_life_expectancy::task_plot_life_expectancy[Asia]>
            └─ <Product example/bld/life_expectancy_asia.svg>
        └─ <Function task_plot_life_expectancy>
            └─ <Dependency example/bld/data.pkl>
            └─ <Dependency example/task_plot_life_expectancy::task_plot_life_expectancy[Europe]>
                └─ <Product example/bld/life_expectancy_europe.svg>
            └─ <Function task_plot_life_expectancy>
                └─ <Dependency example/bld/data.pkl>
                └─ <Dependency example/task_plot_life_expectancy::task_plot_life_expectancy[World]>
                    └─ <Product example/bld/life_expectancy_world.svg>

(epp) → example
```

Run pytask



The screenshot shows a terminal window titled "hmg@hmg-home:~/econ/example". The command "(epp) → example pytask" is run, starting a pytask session. The session summary indicates:

- Platform: linux -- Python 3.11.0, pytask 0.4.0rc3, pluggy 1.3.0
- Root: /mnt/econ/example
- Collected 3 tasks.

A table displays the collected tasks:

Task	Outcome
task_clean_data.py::task_clean_data	.
task_plot_life_expectancy.py::task_plot_life_expectancy[Asia]	.
task_plot_life_expectancy.py::task_plot_life_expectancy[Europe]	.

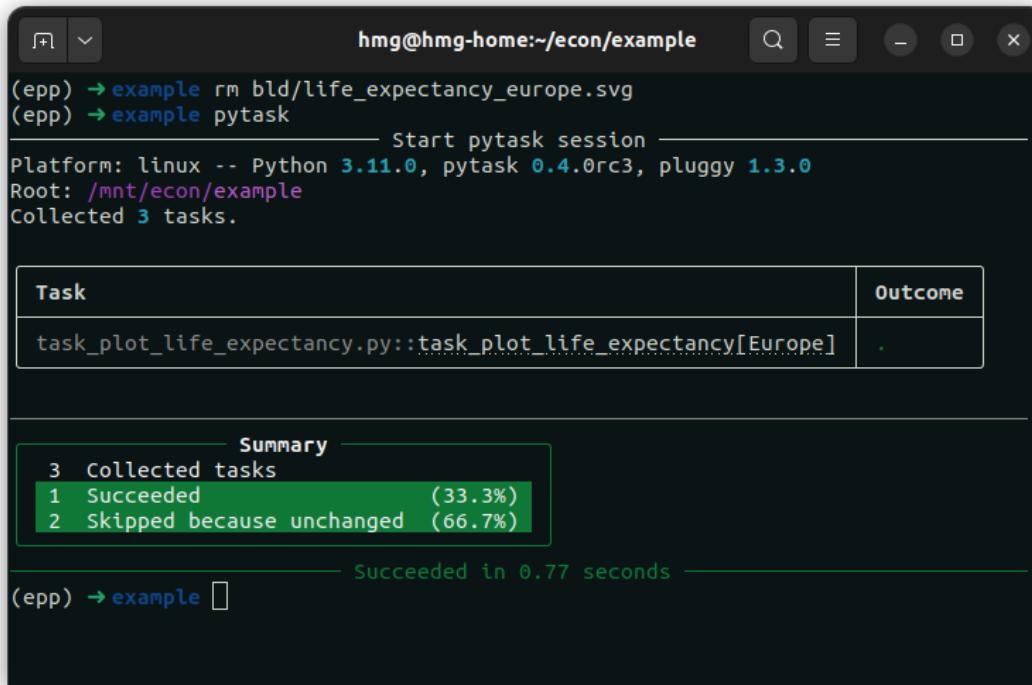
The "Summary" section shows:

- 3 Collected tasks
- 3 Succeeded (100.0%)

The session succeeded in 1.02 seconds.

(epp) → example □

Delete plot and run again



The screenshot shows a terminal window titled "hmg@hmg-home:/econ/example". The user has run the command "pytask" after removing a file named "life_expectancy_europe.svg". The terminal displays the following output:

```
(epp) → example rm bld/life_expectancy_europe.svg
(epp) → example pytask
Start pytask session -
Platform: linux -- Python 3.11.0, pytask 0.4.0rc3, pluggy 1.3.0
Root: /mnt/econ/example
Collected 3 tasks.

Task                                Outcome
task_plot_life_expectancy.py::task_plot_life_expectancy[Europe]  .

Summary
3 Collected tasks
1 Succeeded      (33.3%)
2 Skipped because unchanged (66.7%)

Succeeded in 0.77 seconds
(epp) → example
```

The terminal window has a dark background with light-colored text. It includes standard window controls (minimize, maximize, close) at the top right. The code output is in green, while the terminal prompt and some status text are in blue.

Looping over tasks

- Define your function as usual, but within a loop body
- Set an id based on the running variable(s) via `@task(id=running_var)`
- Set path arguments based on running variable
- Could pass other Python objects, like running variable itself

Looping over tasks or over products?

- Whatever makes your project structure clearer!
- Same style of graphs based on the same dataset: Probably loop over products
- Model specifications: Loop over tasks
- Long running tasks: Loop over tasks
- Looping over tasks yields more granular structure