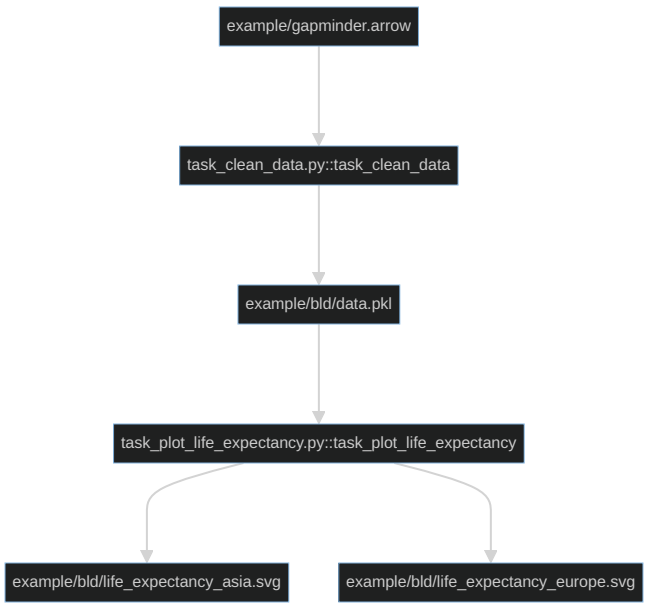
# Effective Programming Practices for Economists

# Reproducible Research

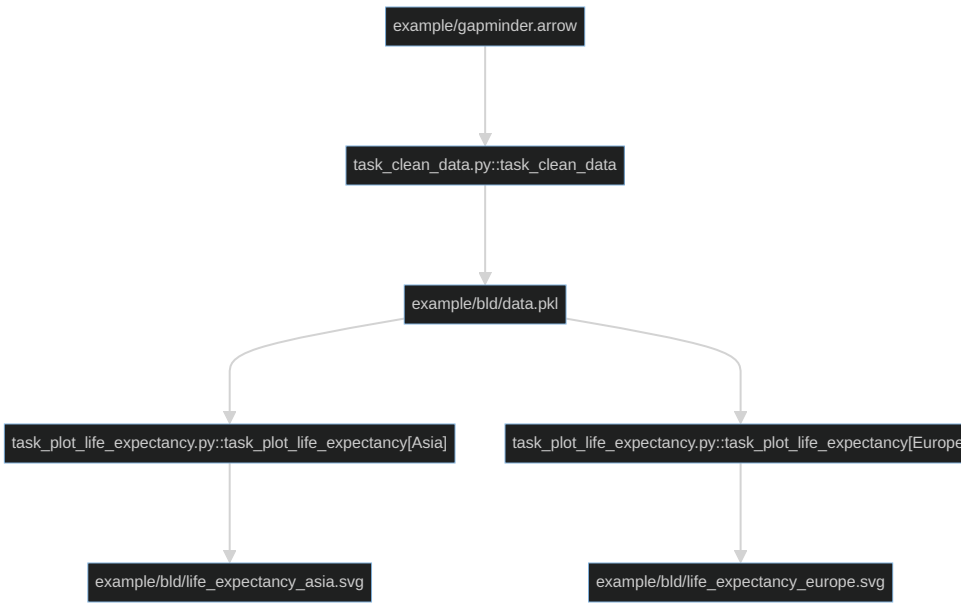
## Re-using pytask functions

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)
```

An error occurred on this slide. Check the terminal for more information.

An error occurred on this slide. Check the terminal for more information.

An error occurred on this slide. Check the terminal for more information.

# 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