# Code pipelines in Python

## Managing code with Make

2025-03-24

# Strategies for reproduciblity


Art by Allison Horst

# Clean execution environments

## *Quarto documents*

````
In this exercise, we'll look at a report generated completely in Python using
Quarto.
We're using an intentionally simplified report: what we're telling the readers,
how we fit models, and so on, are all much simpler than you would use in practice.
Instead, focus on the big idea here: reproducible reports!


▷ Run Cell | Run Next Cell
```{python}
#| label: setup
#| include: false
import polars as pl
import polars.selectors as cs
from great_tables import GT
import pickle
```
````

# Clean execution environments

## *Fresh Jupyter Kernels*

# Strategies for reproduciblity

- **Quarto** for reproducible documents
- **Virtual environments** for package version management, such as **conda** or **uv**
- **git/GitHub** for version control

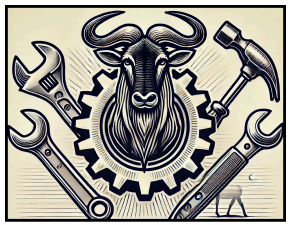
# Script-oriented workflows

```
01-read-data.py
02-clean-data.py
03-decriptive-stats.py
...
n-output.py
report.qmd
```

- Doesn't scale well—in terms of both time and scope

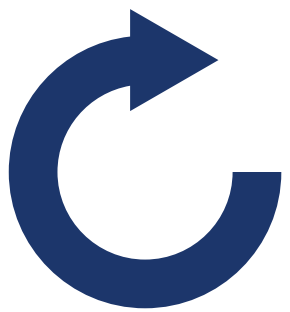
- Not clear what we can skip

# GNU Make



**Scale** the work you need.

**Skip** the work you don't.



**See** evidence of reproducibility.

# Setting up a pipeline

- Make requires a `Makefile` file in the root directory of your project

Makefile

```
figures/gapminder_plot.png: scripts/create_line_plot.py data/gapminder.csv
    uv run scripts/create_line_plot.py
```

```
├── data
│   └── gapminder.csv
├── scripts
│   └── create_line_plot.py
└── Makefile
```

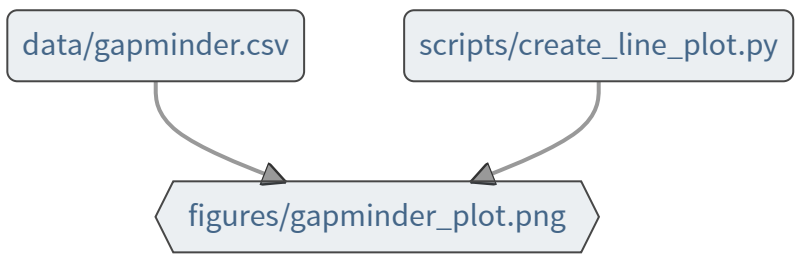
scripts/create_line_plot.py

```
def create_line_plot(df: pl.DataFrame) -> plt.Figure:
    ## -- snip --
    return fig

def main():
    gapminder = pl.read_csv("data/gapminder.csv")
    fig = create_line_plot(gapminder)

    # Ensure the output directory exists.
    os.makedirs("figures", exist_ok=True)

    # Save the figure as a PNG image.
    fig.savefig("figures/gapminder_plot.png", dpi=300, bbox_inches="tight")
    plt.close(fig)

if __name__ == "__main__":
    main()
```

# Visualizing `Makefile`

Makefile

```
figures/gapminder_plot.png: scripts/create_line_plot.py data/gapminder.csv
    uv run scripts/create_line_plot.py
```



# What's Make's plan?

```
make --dry-run
```

```
uv run scripts/create_line_plot.py
```

Run this in the terminal! Don't put it in `Makefile` or other scripts.

# What's Make's plan?

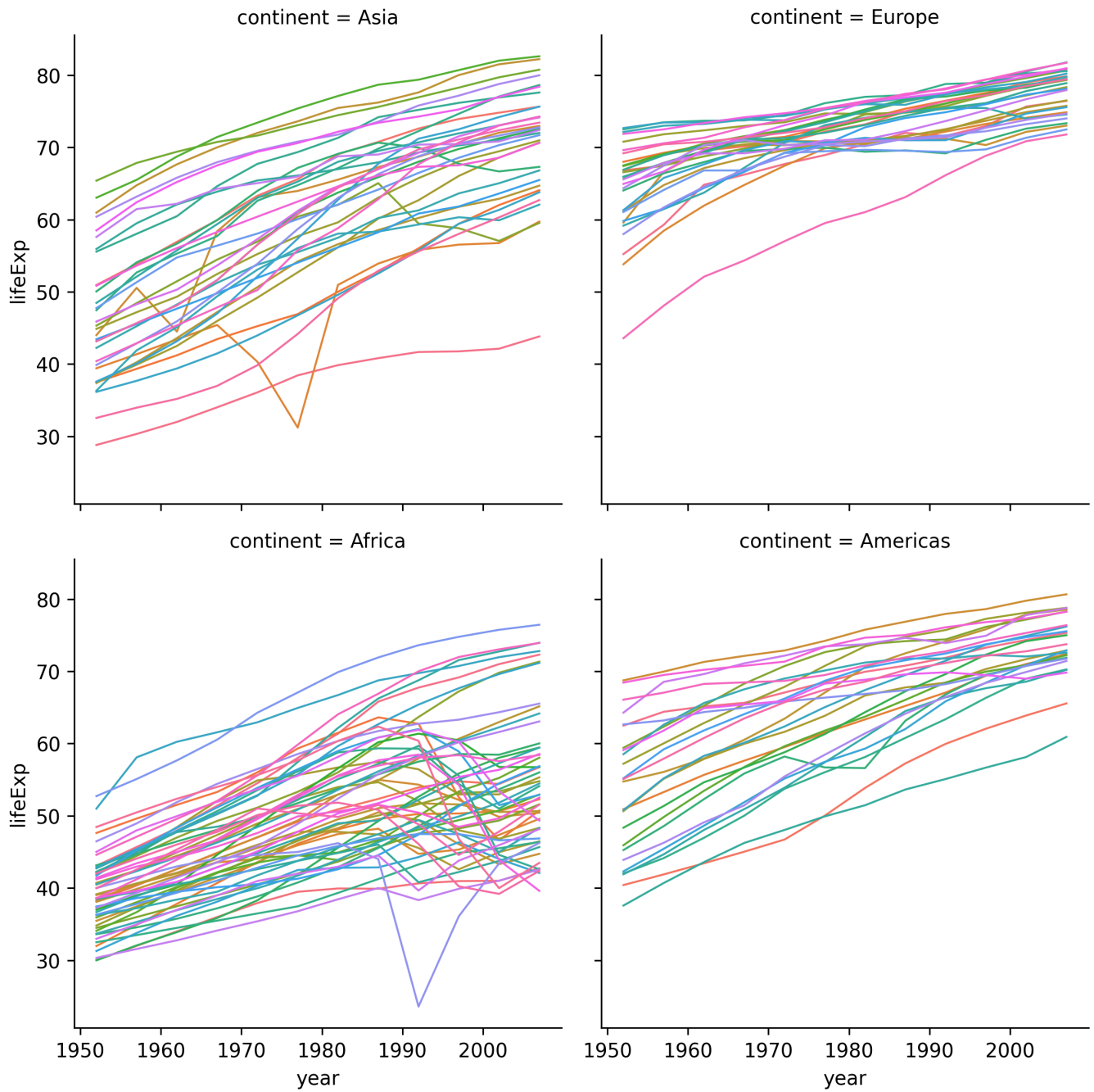
```
1  make -n
```

# Building the pipeline

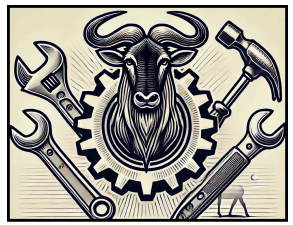
```
make
```

```
uv run scripts/create_line_plot.py
```

```
├── Makefile
├── data
│   └── gapminder.csv
├── figures
│   └── gapminder_plot.png
└── scripts
    └── create_line_plot.py
```

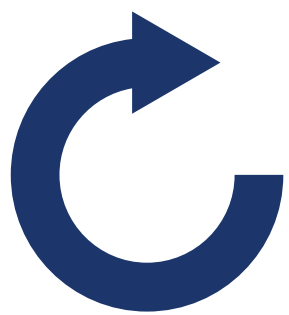

continent = Asia
continent = Europe
continent = Africa
continent = Americas
lifeExp
year
80
70
60
50
40
30
1950
1960
1970
1980
1990
2000

# GNU Make



**Scale** the work you need.

**Skip** the work you don't.



**See** evidence of reproducibility.

# *Your Turn 1*

**Work through *Your Turn 1* in `exercises.qmd`**

# Makefile

```
1 target: dependency
2     recipe
```

The indent *must* be a tab!

# Makefile

```
target: dependency1 dependency2
    recipe
```

# Makefile

```
target1: target2
    recipe1

target2: dependency1 dependency2
    recipe2
```

# Makefile

```
1 all: target2
2
3 target1: dependency1 dependency2
4     recipe1
5
6 target2: target1
7     recipe2
```

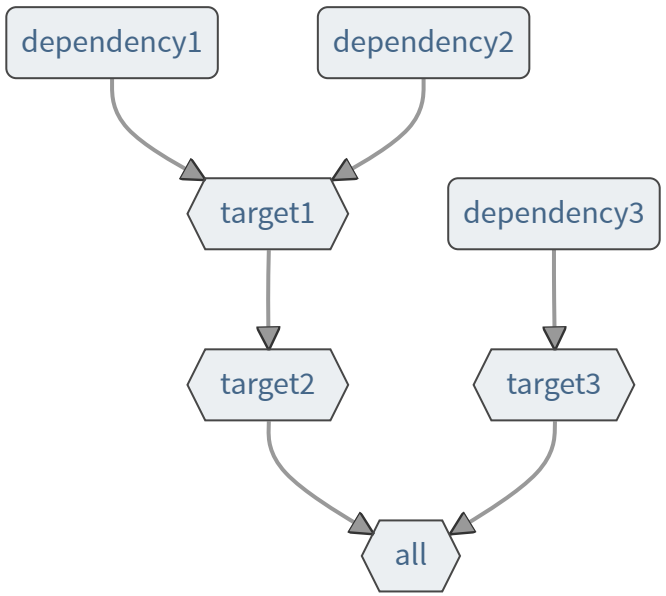
# Makefile

```
all: target2 target3

target1: dependency1 dependency2
    recipe1

target2: target1
    recipe2

target3: dependency3
    recipe3
```

# Makefile

```
1  .PHONY: all
2  all: target2 target3
3
4  target1: dependency1 dependency2
5      recipe1
6
7  target2: target1
8      recipe2
9
10 target3: dependency3
11     recipe3
```

# Makefile


dependency1
dependency2
target1
dependency3
target2
target3
all

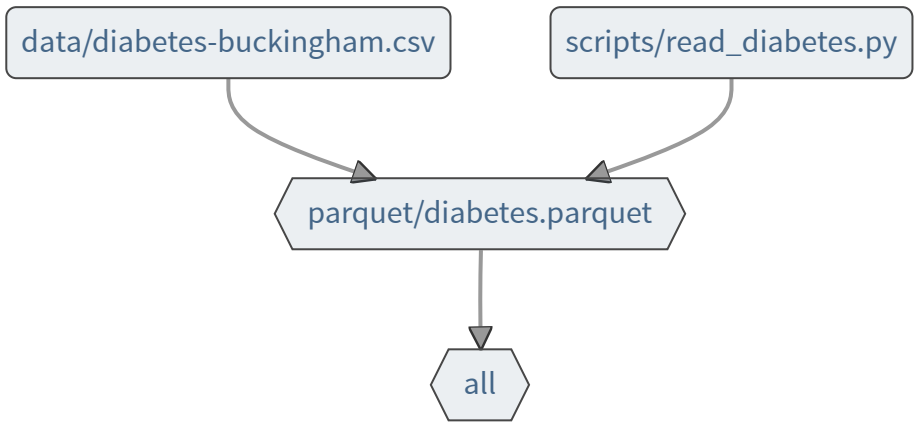
# Makefile: make clean

```
1  .PHONY: all clean
2  all: target2 target3
3
4  # --snip--
5
6  clean:
7    rm -f target1 target2 target3
```

# *Your Turn 2*

**Work through Your Turn 2 in `exercises.qmd`**

# Your Turn 2


data/diabetes-buckingham.csv
scripts/read_diabetes.py
parquet/diabetes.parquet
all

# Re-running targets

```
make
```

```
make: `figures/gapminder_plot.png' is up to date.
```

scripts/create_line_plot.py

```python
def create_line_plot(df: pl.DataFrame) -> plt.Figure:
    ## -- snip --
    g = sns.FacetGrid(
        df,
        col="continent",
        col_wrap=2,
        height=4,
        sharey=True,
        hue="country",
        palette="mako"
    )
    ## -- snip --
    return fig

## -- snip --
```

# Outdated targets

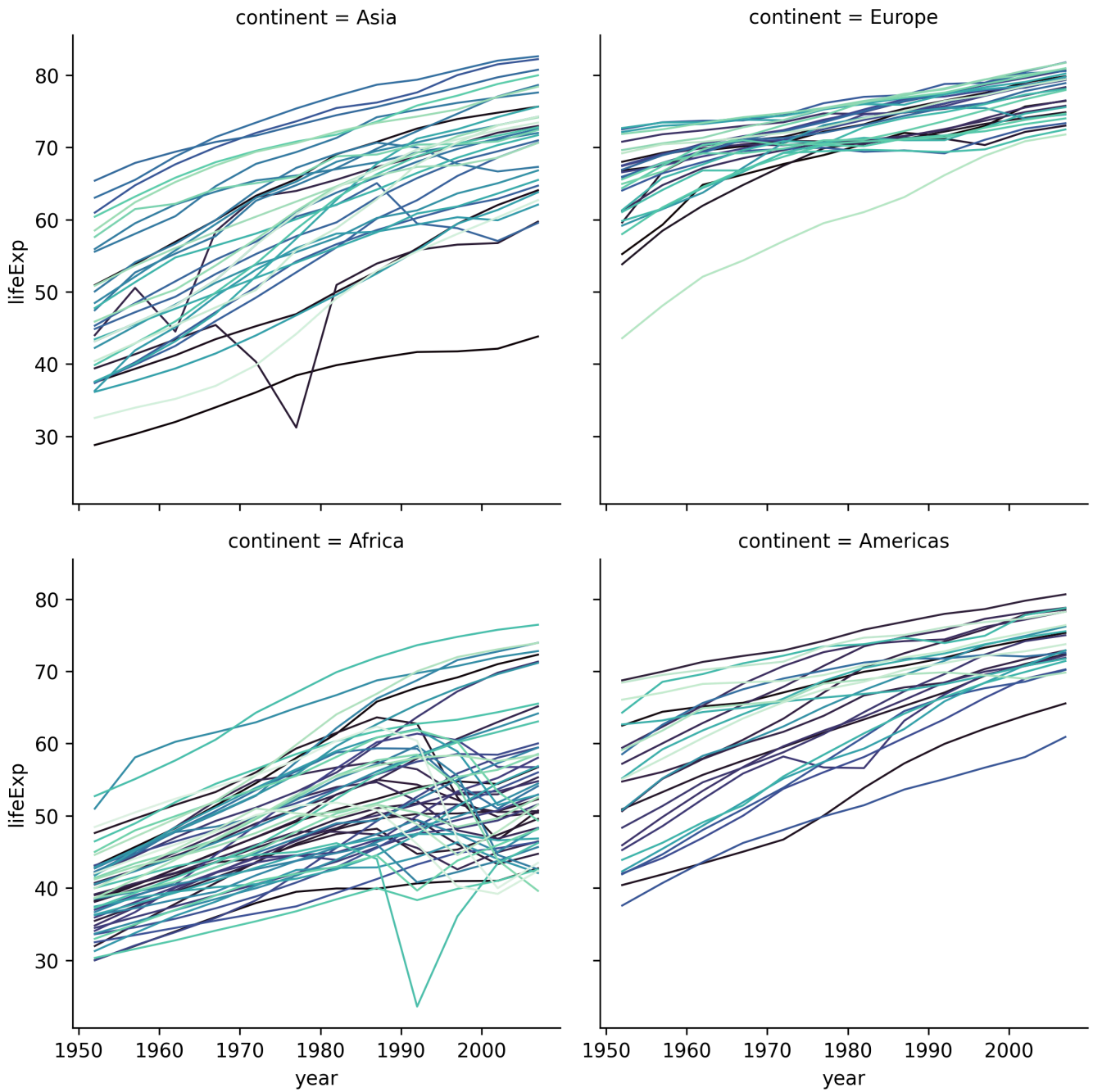
```
make --dry-run
```

```
uv run scripts/create_line_plot.py
```

# Outdated targets

```
1 make
```

```
uv run scripts/create_line_plot.py
```


continent = Asia
continent = Europe
continent = Africa
continent = Americas
lifeExp
lifeExp
80
70
60
50
40
30
1950
1960
1970
1980
1990
2000
year
year

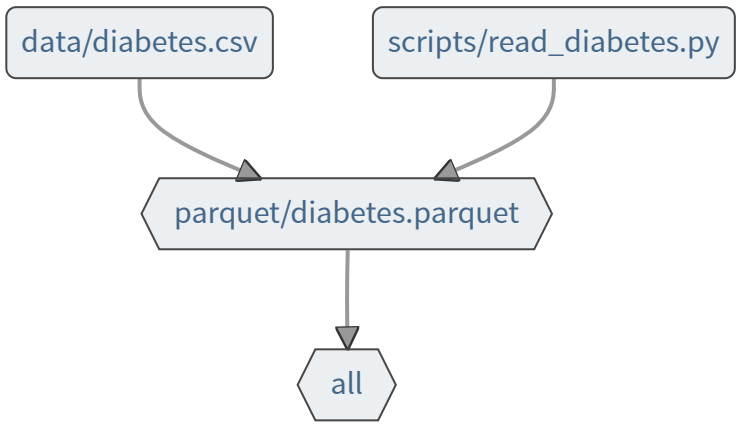
# *Your Turn 3*

Change the `parquet/diabetes.parquet` target to depend on `diabetes.csv` instead. Also modify `scripts/read_diabetes.py` to use this file.

Predict which targets are going to re-run, then run `make -n`. Were you right?

Run `make`

# Your Turn 3


data/diabetes.csv
scripts/read_diabetes.py
parquet/diabetes.parquet
all

# Building up your pipeline

- *Good targets are meaningful units of your analysis or important dependencies like files*
- *Add one or two targets at a time*
- *Run `make -n` and `make` often*

# *Your Turn 4* and *Your Turn 5*

## Work through *Your Turn 4* and *Your Turn 5* in `exercises.qmd`

TODO: add mermaid for 4 and 5

# Including Quarto files as targets:

## `report.qmd`

````
```{python}
#| label: setup
#| include: false
import polars as pl
gapminder = pl.read_csv("data/gapminder.csv")
```

These data have `{python} len(gapminder)` observations.

```{python}
#| label: fig-one
#| fig-cap: "Figure 1"
import IPython.display as disp
disp.Image("figures/gapminder_plot.png")
```
````

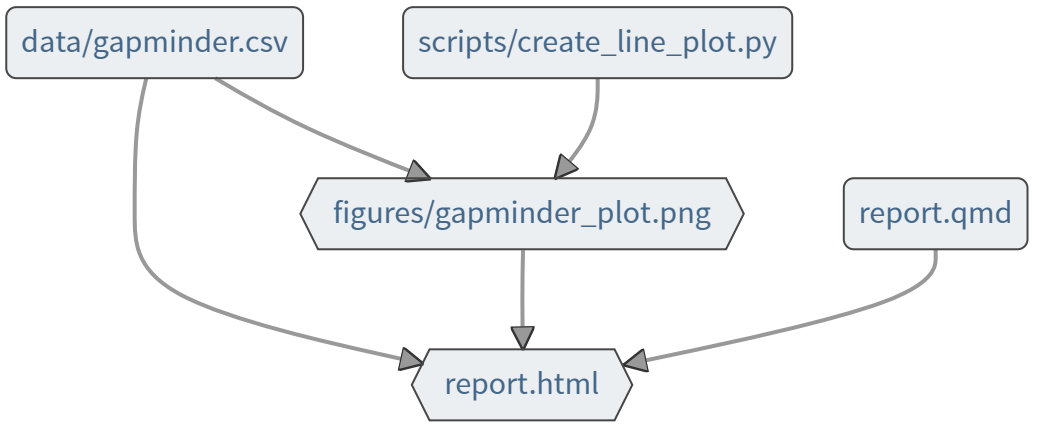
Makefile

```
report.html: report.qmd figures/gapminder_plot.png data/gapminder.csv
    uv run quarto render report.qmd

figures/gapminder_plot.png: scripts/create_line_plot.py data/gapminder.csv
    uv run scripts/create_line_plot.py
```

# make

```
uv run quarto render report.qmd
```

# Including Quarto files as targets



# *Your Turn 6*

## Work through Your Turn 6 in `exercises.qmd`

TODO: mermaid

# Removing target files

- **make clean**: A custom but common command to remove all created targets. Resets the pipeline to scratch.

# *Your Turn 7*

**Confirm that you can reproduce your entire pipeline from scratch. In the terminal:**

**Run `make clean`**

**Run `make -n`**

**Run `make`**

# Automatic parallelization

```
make --jobs=N
```

# Automatic parallelization

```
1 make --jobs=4
```

# Automatic parallelization

```
1  make -j4
```

# Wildcards

```
.PHONY: all clean

# Find all CSV files in the data/ directory.
CSV_FILES := $(wildcard data/*.csv)
# Generate a list of Parquet file names in the parquet/ directory.
PARQUET_FILES := $(patsubst data/%.csv,parquet/%.parquet,$(CSV_FILES))

# Default target: build all Parquet files.
all: $(PARQUET_FILES)

# Pattern rule: convert each CSV file to a Parquet file.
parquet/%.parquet: data/%.csv
    uv run scripts/convert_csv_to_parquet.py $< $@

clean:
    rm -f parquet/*.parquet
```

# Resources

## The GNU Make Manual

## The Software Carpentry Make Course