Data Preparation and Workflow Management

Hannes Datta

Pipeline Automation using `make`

Motivation & learning objectives

- Directory and file chaos, can't find my work
  - Let's **organize** files and directories
- Difficulties (re)executing the project
  - Let's define a project pipeline, and **automate** it fully

Disclaimer

- this is just *one* way to set up a project (many alternative directory structures and build tools exist)
- project setup varies with project characteristics (e.g., degree of collaboration, necessity to run code on multiple machines)
- here: chosen a relatively simple setup on a local computer
- start *doing* it

How to organize your project?

1. Create project folder & structure

Each project consists of data, source code, and generated temporary and output files. Store them separately.

```
\data         <- your raw data, unmodified
\src          <- any source code
\gen\temp     <- any files that are not really needed later
\gen\output   <- any "final" files produced
\gen\audit    <- any files that allow you
                 to "check" your work
```

2. Define & create pipeline stages

Think about the common steps that will be done in your project, for example:

1) **Prepare** dataset for analysis

2) **Analyze** dataset using a statistical model
3) Produce **paper**

Create directories for each pipeline stage

```
\src\datapreparation
\src\analysis
\src\paper
\gen\datapreparation\temp\
\gen\datapreparation\output
\gen\analysis\temp\
\gen\analysis\output
\gen\paper\temp\
\gen\paper\output
```

3. Plan each pipeline stage

- In each stage of the pipeline, **multiple source code files** will "produce" an output
- Output varies per pipeline stage
  - e.g., dataset, analysis, PDF report, launch of an App, populating a dashboard

Example: Data preparation

- Download public datasets (used for some control variables),
- Load other data sets from various sources (e.g., from Excel)
- Merge primary dataset with control variables
- Generate derivative (aggregate) datasets on a weekly and monthly level
- Save final datasets
- Audit data using RMarkdown

Example: Analysis

- Load final datasets from previous pipeline stage
- Estimate models on both datasets
- Systematically compare both models
- Choose the best model, and save it

Example: Paper

- Load final analysis results from previous pipeline stage
- Produce tables and figures for paper
- Produce slide deck

Visualization (of a different example)

Examples pipelines

Start building your pipeline

- Move existing code to pipeline stages, or write new code in pipeline stages (`src\`)
- Breaks up a long make-all-the-things scripts into discrete, manageable chunks (i.e., components).
    - inputs
    - transformations
    - outputs

Advantages of using a pipeline

1. Structure documents the workflow, making communication with colleagues (and your future self) more efficient
2. When you modify one stage of the pipeline, you don't have to rerun the entire pipeline (e.g., save processing resources).
3. Eventually reproduce entire workflow

4. Automate your pipeline

- What's automation?
    - A set of rules (instructions for the computer)
    - We write them in `makefiles`
- Each rule consists of
    - what to build? ("target")
    - what do I need to build it? ("prerequisites"), and
    - how to build it? (i.e., commands)
- `Make` then builds project

A simple `make` rule

Syntax:

```
targets: prerequisites
   command
   command
   command
```

Example

```
../../gen/data-preparation/aggregated_df.csv: ../../data/listings.csv ../../data/reviews.cs
    Rscript clean.R
```

Example

Phony targets & stitching things together

- Targets that do not produce any output
- Convention
  - `all` = call all targets
  - `clean` = remove all generated files

```
all: one two

one:
    touch one.txt
two:
    touch two.txt

clean:
    rm -f one.txt two.txt
```

Variables

```
INPUT_DIR = src/data-preparation
GEN_DATA = gen/data-preparation

$(GEN_DATA)/aggregated_df.csv: data/listings.csv data/reviews.csv
  $(INPUT_DIR)/clean.R
```

Summary

1. Create directory structure (`src`,`gen`,`data`)
2. Create pipeline stages (e.g., `datapreparation`, `analysis`)
3. Move source code to pipeline stages, or start writing new source code. Break up code in small code chunks rather than one giant script.
4. Automate each pipeline stage