Code, Data, & Version Control: Best Practices for Economic Research

Brendan M. Price
Federal Reserve Board

May 2022

Email: brendan.m.price@frb.gov. Website: www.brendanmichaelprice.com.

Best practices for reproducible research

Two interrelated topics:

- "Best practices"
- Effective workflow

What I want to share:

- High-level principles
- Practical tips

Plan for today

These slides:

1. Motivation
2. Files & folders
3. Code & data
4. Version control

Ask questions!

MOTIVATION

Why bother?

There are always reasons to *neglect* best practices

- Looming deadlines
- "I'll fix it later …"

Effective workflows do entail upfront costs

But the costs are dwarfed by the benefits

- Better research *process*
- Better research *product*

Good workflow aids the research *process*

It saves time

- Less time looking for files
- Less time debugging
- Less duplication of effort

It facilitates analysis

- Rapid prototyping
- Ease of exploration

It feels good

- Less frustration
- Less panic
- More pleasure in the craft

Good workflow aids the research *product*

It encourages good science

- More discoveries
- Fewer mistakes

It complements presentation

- Better figures & tables
- Ease of answering questions

It has positive spillovers

- Shareable code
- Replication packages

All of which confers professional credibility

FILES & FOLDERS

A new project

You're starting a new project

- Might be a solo project
- Might be joint

You have some initial leads

- A question you want to answer
- Data you want to explore

How should you organize your files?

The project directory

Give the project its own directory

- Completely self-contained
- No external cross-dependencies

Organize it coherently

- Separate files by function
- Choose clear & concise names
- Exploit parallel structure
- Avoid redundancy
- Minimize clutter

Ensure derived files are traceable back to whatever created them

A battle-tested approach

Four (or more) subdirectories:

- CODE
- DATA
- LOGS
- OUTPUT

Sometimes add a few more

- MODELS
- PACKAGES
- PAPER
- SLIDES

A battle-tested approach (continued)

Subdivide CODE by function

- CODE/BUILD (data preparation)
- CODE/LEARN (exploratory analyses)
- CODE/SHARE (external-facing analyses)

Subdivide DATA by provenance

- DATA/RAW (data as provided to you)
- DATA/DERIVED (anything you created)

Use parallel structure & recycle names

- CODE/CLEAN_DATA.DO $\Longrightarrow$ LOGS/CLEAN_DATA.LOG
- CODE/LEARN/TIMEUSE.DO $\Longrightarrow$ OUTPUT/LEARN/TIMEUSE.PDF

Use (only) as much hierarchy as you need

Hierarchy should scale with project complexity

- Simple project $\Longrightarrow$ “flat” directory structure
- Complex project $\Longrightarrow$ subdirectories, subsubdirectories …

Start simple, elaborate as needed

CODE & DATA

Reproducibility

The "codebase" maps inputs into outputs

- Inputs: raw data
- Outputs: figures, tables, & findings

Reproducibility: rerunning code yields identical output

- At least on your computer
- Ideally on my computer, too

Gold standard: "one-click execution"

- Requires a program like MAIN.DO
- Facilitates experimentation

Pitfalls: artifacts, external dependencies, operating systems

Desiderata

Aside from reproducibility, good code strives for:

- Brevity
- Readability
- Robustness
- Maintainability
- Efficient runtime
- Efficient storage

Sometimes these goals conflict

But usually they're complementary

- Concise code usually runs faster
- Readable code is easier to maintain

Coding tips

To achieve these goals:

- Automate extensively
- Comment extensively
- Test your code
- Refine your code
- Learn new tricks

Time spent improving code usually pays for itself

Managing data

Cardinal rule: never overwrite raw data

- Record when/where you got the data
- Leave raw extracts 100% unmodified

Operate through code, not manually

- Fine to experiment interactively
- But “real” work happens in scripts

Save intermediate files only when necessary

- Usually just clutter, costly storage
- Look for workarounds

VERSION CONTROL

What is version control?

Version control: a systematic record of revisions to a set of files

- User saves "snapshots" of code *&* code-like files
- Easy to recover code from any given snapshot
- Saves storage space
- Avoids clutter

Industry standard is Git

- Usable on its own
- But usually paired with GitHub (or GitLab)

Learn Git . . . eventually

Git is well worth learning

- Hugely helpful for writing a dissertation
- Widely used in academia, policy, industry

But learning curve is a bit steep

So: start slow *&* stick with it

- Don't expect to understand it all at once
- Learn it in bits *&* pieces
- With time *&* practice, it's a game-changer

Lots of good resources online

CONCLUDING THOUGHTS

The big picture

Main message: think (hard) about workflow

Invest early in good habits

- Be organized
- Find ways to improve
- Figure out what works for you

Don't go it alone

- Talk to your classmates
- Read people's code
- Get the advice you need

Recommended reading: Gentzkow & Shapiro (2014)

Good luck!