File Organization

Jonathan Tannen

Agenda

- Better Engineering: File Organization
- *10 minute break*
- Proposal and Data Presentations

Better Engineering?

Code is for *humans*.

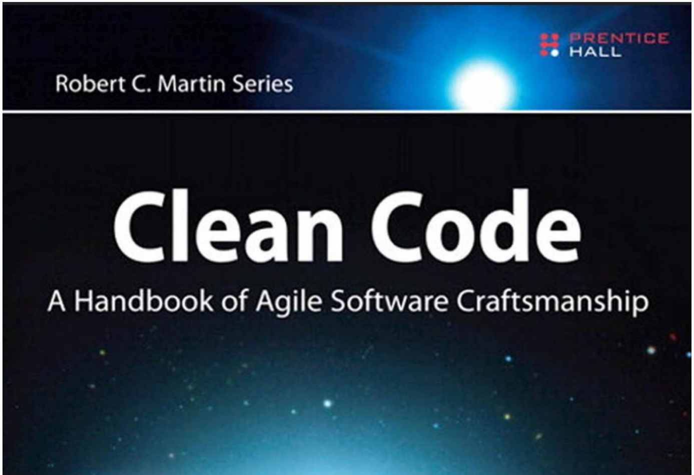
Good code:

- Is easy to understand and navigate.
- Is obvious how to extend.
- Minimizes time needed to make a change.
- Minimizes opportunity for mistakes.

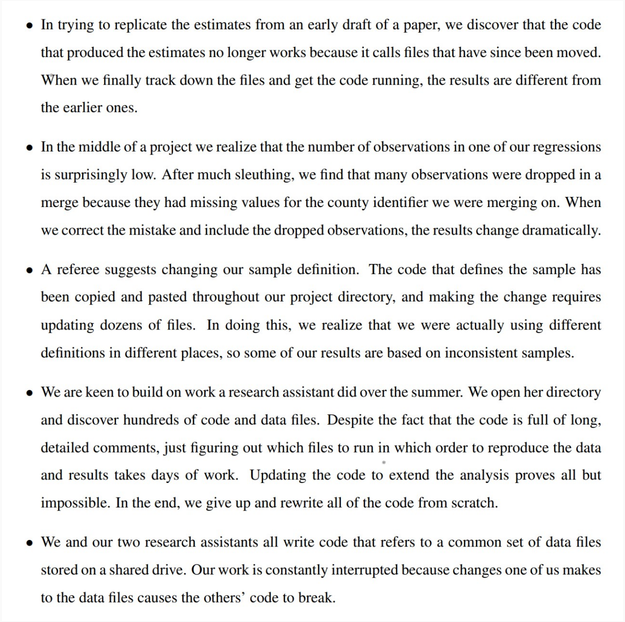
Today: Data Cleaning and File Organization

Inspired by:

- Code and Data for Social Sciences, Gentzkow & Shapiro
- Clean Code, Martin



What can go wrong? (from Gentzkow & Shapiro)



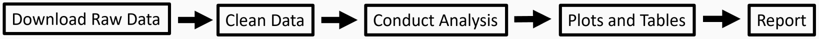
- In trying to replicate the estimates from an early draft of a paper, we discover that the code that produced the estimates no longer works because it calls files that have since been moved. When we finally track down the files and get the code running, the results are different from the earlier ones.
- In the middle of a project we realize that the number of observations in one of our regressions is surprisingly low. After much sleuthing, we find that many observations were dropped in a merge because they had missing values for the county identifier we were merging on. When we correct the mistake and include the dropped observations, the results change dramatically.
- A referee suggests changing our sample definition. The code that defines the sample has been copied and pasted throughout our project directory, and making the change requires updating dozens of files. In doing this, we realize that we were actually using different definitions in different places, so some of our results are based on inconsistent samples.
- We are keen to build on work a research assistant did over the summer. We open her directory and discover hundreds of code and data files. Despite the fact that the code is full of long, detailed comments, just figuring out which files to run in which order to reproduce the data and results takes days of work. Updating the code to extend the analysis proves all but impossible. In the end, we give up and rewrite all of the code from scratch.
- We and our two research assistants all write code that refers to a common set of data files stored on a shared drive. Our work is constantly interrupted because changes one of us makes to the data files causes the others' code to break.

What is the answer?

Not working harder. Working smarter.

Rules for Data Analysis

Rule 1: Modularity. Separate your workflow into steps. Each step gets its own script(s). Save outputs in between.



My original workflow

1. Download raw data
2. Clean it by hand
3. Gigantic script that reshaped the data, ran analysis, generated plots.
    a. Hard-coded assumptions hidden everywhere including settings, file locations, dates.
    b. This would often break. Only way to check was post-hoc, or error debugging.
4. Write report by hand. Copy numbers from printed output.

What does this mean?

- If something needs to change. . .
  1. Only I can do it.
  2. There are five different places that need to change. I will forget some of them.
  3. It takes a long time.
  4. Very high probability of breaking something or (worse) introducing correctness error.

What is the answer?

Modularity

Also known as “Separation of Concerns” or “Single Responsibility Principle”.

“Separate an application into steps, with minimal overlapping functions between them and clear deliverables in between.”

Modularity of data analysis

1. Download raw data.
2. Clean the data.
3. Conduct analysis.
4. Produce plots and tables.
5. Create report.

These should be separate scripts that save results. The next script only depends on the saved outputs of the last one.

Why does this help?

Suppose you get next year's raw data, but the format has changed.

All you need to do is munge the data to the same format as the last clean output!

This format means

a. The expectations are clear.
b. You won't accidentally mix in assumptions later on.

A note

None of this is language specific, or even specific to coding.

Rules for Data Analysis

Rule 2: Automation. Automate everything that can be automated.

Why automate?

1. Reproducible
2. Auditable
3. Reduces human error

Non-obvious things to automate

1. Manually fixing data errors.
2. Copying files between locations.
3. Copying text from an output to a report.
4. Running scripts in the right order.

Ask yourself: “As I work through this, what is manual? How many clicks are required?”

Raw Data

Rule 2a: Keep your raw data in a separate file. You *never* modify this directly.

Confident Automation

Produce plots and outputs that make you confident in the results.

- Counts of data.
- Means and distributions.
- Maps and summary plots.
- (Eventually) Unit tests.

What do you look at when something goes wrong? Automate that!

Zooming out

Don't be overwhelmed! Do better than yesterday. Your future self will thank you.

Zooming out

There will always be human interaction needed. But ask yourself, what's the current bottleneck of my time? The current riskiest part if someone else tried to run this?

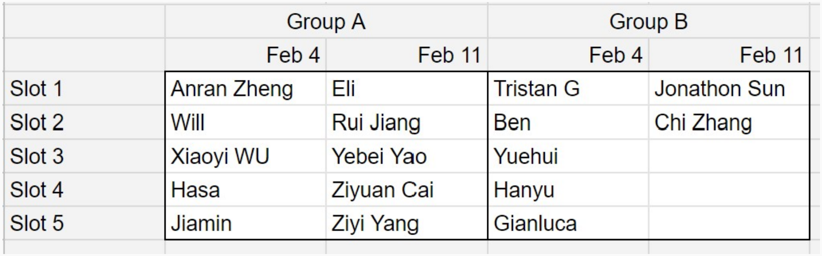
And now. . .

Live Demo!

Pre- sen- ta- tions - A 7 minute pre- sen- ta- tion (with 3 min

Presentations

- 10 minute break.
- Presentations for 1 hour.
- See you back here at XX:XX
- Group A: Here
- Group B: Room 323



	Group A		Group B	
	Feb 4	Feb 11	Feb 4	Feb 11
Slot 1	Anran Zheng	Eli	Tristan G	Jonathon Sun
Slot 2	Will	Rui Jiang	Ben	Chi Zhang
Slot 3	Xiaoyi WU	Yebei Yao	Yuehui	
Slot 4	Hasa	Ziyuan Cai	Hanyu	
Slot 5	Jiamin	Ziyi Yang	Gianluca	

Presentation Debrief

1. What did you learn from someone else's presentation?
2. What good feedback did you get?
3. What are your next open questions?
4. Any common themes across projects?

Next week

1. Presentation from Seth Bluestein and Michelle Montalvo!
2. Second batch of presentations.