Reproducible Coding Strategies

Garret Christensen[1]

[1]UC Berkeley:
Berkeley Initiative for Transparency in the Social Sciences
Berkeley Institute for Data Science

IDB, March 2018
Slides available online at
http://www.github.com/BITSS/IDBMarch2018

Outline

BERKELEY INITIATIVE FOR TRANSPARENCY
IN THE SOCIAL SCIENCES

The Claerbout Principle:

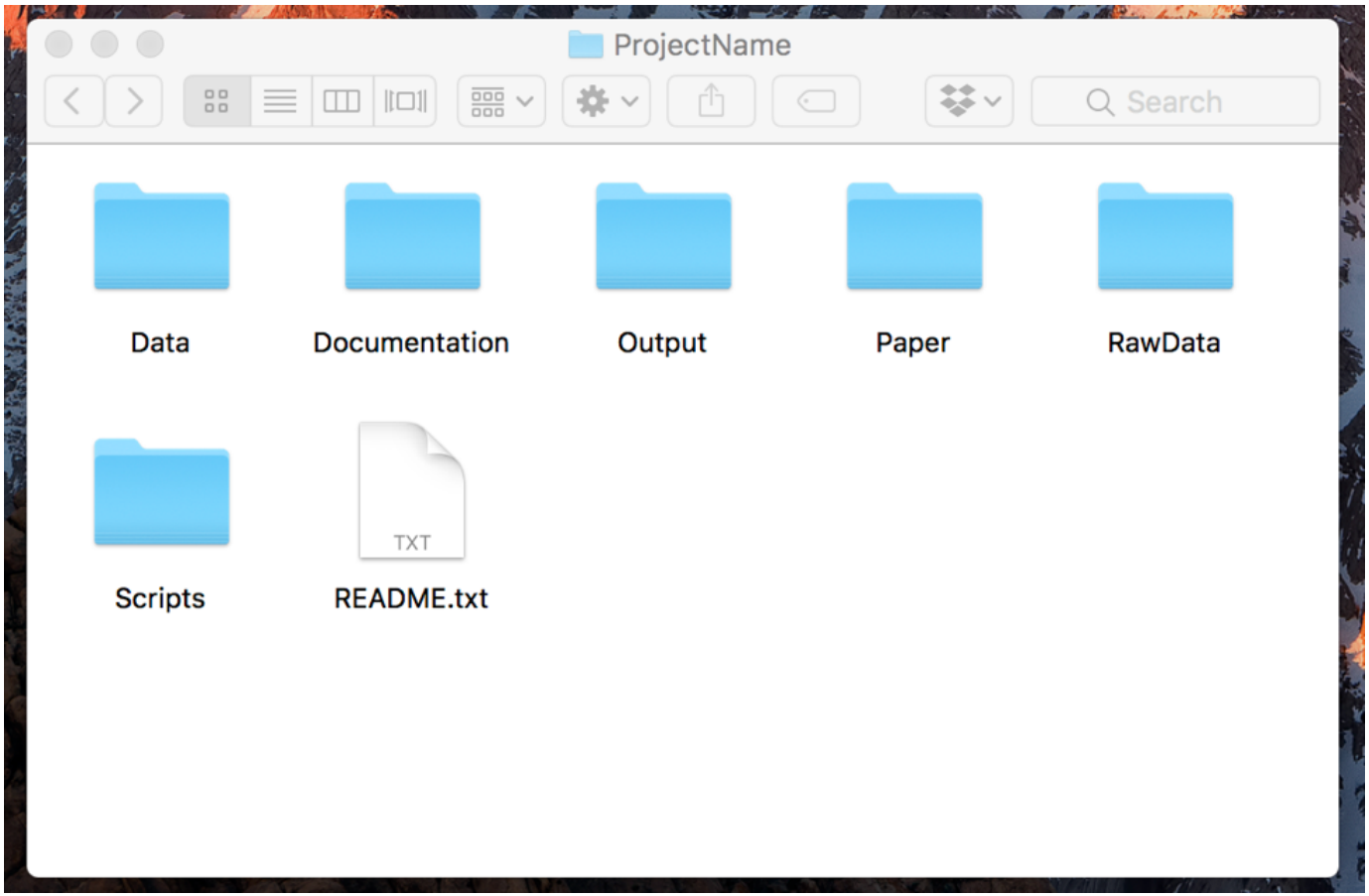
> *"An article about computational science in a scientific publication is not the scholarship itself, it is merely advertising of the scholarship. The actual scholarship is the complete software development environment and the complete set of instructions which generated the figures."*
> *(Buckheit & Donoho, 1995)*

Organizing Principles

1. Use code (scripts), don't work by hand (Excel/spreadsheet, GUIs).
2. Consider not saving statistical output, and just saving the code and raw data that generates it.
3. Reproducibility–on your own machine across multiple runs, across machines, across researchers.

File Management & Coding Suggestions

Begin with a logical file structure



General Coding Suggestions

- Make sure script files are self-contained: don't write code that only works if you run a group of other files previously in a specific order and then leave things hanging precariously.
- Include tests in your code. This can alert you if output changes.
- You can never comment your code too much. Truly explain rather than transliterating: `x=1` as "initialize the population count to 1" or "set x equal to 1."
- Indent your code.

General Coding Suggestions II

- Once posted, any changes at all require a new file name. Better: use version control.
- Separate your data cleaning and analysis files. Don't make any new variables that need saving (or will be used by multiple analysis files) in an analysis file. It is far better to only create a variable once so you know that it is identical when used in different analysis files.

General Coding Suggestions III

- Name variables informatively: pick a side for indicator variables “dead" (or “living”) instead of “status". (gender, race, etc.)
- Don’t leave clutter around-delete temporary or unnecessary intermediate objects.
- You can use a prefix such as `x_` or `temp_` so you know which files or variables can easily be deleted later. Stata also has the `tempfile` and `tempvar` functionality.
- Every variable should have a label. (If allowed for by the program.)
- Use relative directory paths (such as “./Data" not “C:/Users/garret/Documents/Project/Data")

Stata Suggestions

- Accurately and concisely capture missing values. (‘.‘ and ‘.a-.z‘)
- Make sure code always produces the same result, and that merging and sorting is reproducible. ‘duplicates report; isid; sort, stable‘
- Run tests to alert yourself when results change.

Example Test

```
count if _merge!=3
if r(N)!=74 {
display "Unmatched observations changed!"
there is an error here
}
```

- Don’t use abbreviations for variables, or commands beyond reason
- Use global macros to define directory paths so collaborators can readily work across different computers.
- Use local macros for varlists.

Stata Suggestions

- Use computer-stored versions of numerical output (eg ‘r(mean)‘). Use ‘return list‘ and ‘ereturn list‘
- If you have a master .do file that calls other .do files, which each have their own .log file capturing output, you can run multiple log files at the same time (so you end up with a master .log file)
- Use the ‘label data‘ and ‘notes‘.
- Use the ‘notes‘ command for variables as well for identifying information that is too long for the variable label.

Stata Suggestions

- Validate data sources to ensure consistency. Use ‘datasignature‘ on auto data set (‘sysuse auto.dta‘, then ‘datasignature set‘ should give you this number: ‘74:12(71728):3831085005:1395876116‘)
- Use value labels for all categorical variables. ‘numlabel [lblname-list], add command.‘
- Don’t use capital letters in variable names.
- Make your files as non-proprietary as possible (use the ‘saveold‘ command)

Questions?

Thank you!