Seven Suggestions for Best Practices in Data Collection

Philip A. Schrodt

Parus Analytical Systems
schrodt735@gmail.com

March 29, 2014

1. Version control

Data sets change as additional cases and bugs are uncovered.

Data set will necessarily be released before they are perfect because they will never be perfect.

Storage is very very cheap now.

Formal version control systems such as GitHub are awkward to learn but may still be worth doing. But at the very least have a systematic way of keeping backups.

Provide a versioned method of citing your data.

2. Automate as much of the process as possible

In a large system, this reduces costs. In any system, it improves quality control and transparency.

Resist the temptation to say "I'll do this by hand because I'm only going to do it once": you won't do it once.

There are a *lot* of tools available now for this, both proprietary, open access (Google) and open source.

If you are writing a system for machine-assisted human coding, your system should work through a web browser, not dedicated software.

3. Coders selected for convenience are very expensive

As a recovering academic now in the private sector, I'm observing that both GRAs and UGRAs are, with very few exceptions, an extraordinarily bad deal from an economic perspective. Particularly GRAs.

Recruiting coders on the web—decentralized, not necessarily crowd-sourced—is likely to be less expensive. Committed coders provide better quality and consistency.

I am very skeptical about data produced "for free" in class assignments: anything that can be coded by uncompensated and usually poorly supervised undergraduates should be crowd-sourced.

4. Store everything in a text format

Binary formats—anything you can't read—become unavailable over time. Even Microsoft's. Someone will want your data in thirty years.

Which text format probably isn't important: we don't know what the future standards will look like anyway, and programs to translate from common existing formats (e.g. .csv, XML) will undoubtedly be available.

Self-documenting formats are a good thing, though no single standard has caught on.

5. Code in a framework that is compatible with existing ontologies

They don't have to be the same—I'm giving up trying to get you COW people to use ISO-3166—but they need to be easily translated (e.g. *R* `countrycodes` package or `CountryInfo.txt`).

Coding systems that are almost but not quite compatible impose a huge burden on future users.

6. The data generation process must be 100% transparent to protect against mistakes and fraud

> *To err is human; to really screw up requires a computer*

This is a general issue in the sciences—a major new case appears almost weekly now—and not just political science.

Thanks to Gary King's unwelcome hectoring on replication, political science has long-established (though rarely enforced) standards on this which other disciplines are only now finding are necessary.

There are simply too many ways to make mistakes in a complex analysis—they are all complex—not to require full transparency.

Computational tools also make fraud much more easy—as the natural sciences are discovering—and 100% transparency is the only solution.

7. The data generation process must be 100% open source to protect against violations of trust

All successful systems accumulate parasites

Trusted open source networks provide incredible leverage and productivity.

Like any trust network, some individuals—and particularly large organizations—will try to exploit trust, taking from the system and contributing nothing in return: if you haven't learned about the D/C strategy in iterated prisoner's dilemma uh. . . really?

I'd like to say people/organizations don't do this because they are evil, just because they are human, but in fact, they do it because they are evil.

RESIST! And ruthlessly punish defectors, just like the theory tells us to do.


Organize

Thank you

Email: schrodt735@gmail.com

Slides:
`http://eventdata.parusanalytics.com/presentations.html`

Software: `https://openeventdata.github.io/`