Notes for Efficient Data Organization/Handling

Andrew K. Rose

Updated: July 31, 2017

Precision is *always* more important than speed.

File Folders

- Each project should have a directory with a short descriptive name.

    1. There should be three sub-directories off that, “Progs” for storage of programs and output;“Data” for raw and massaged data sets and “Docs” for documents (paper drafts, slides, readings, notes, etc.)

Data

- Hard disk space is basically free: use it!
- Do your data massaging in programmed steps, as modular as possible. That way if you make a mistake, you can fix the program and then regenerate the data easily.
- Each excel spreadsheet should have a “Read Me” sheet at the outset. This should list the original sources of the data, including URLs and the date things were originally downloaded. Also, as you move the data through various steps (e.g., to cut out data headers, change variable names etc), use a number of different clearly labeled sheets in the same workbook and describe the steps in the “Read me” sheet.
- When massaging data, leave the data raw in at least one easily accessible format. For instance, if you’re handling data in Excel, leave it raw in one sheet, and do your massaging on a copy of that.
- Never do something by hand when you can program it. Programs can be replicated, modified, checked, shared, and usually take less time to do something with far greater precision.
- Always leave your data sets as raw as possible; do the transformations inside the program.
- Check your data sets by graphing and looking for outliers. Look for outliers through descriptive statistics (e.g., look for stuff that lies a few standard errors beyond the mean). Time-series plots (of both levels and growth rates) are great for time-series.

- Always check your data set thoroughly before you do anything that might make you stop the checking (looking for results prematurely can make you either give up too soon or find something that's simple a data mistake).
- Label all your variables.
- Create all scratch files with a common prefix (I use crap*) and delete them at the end of the program.
- Create your data sets such that they can be posted. This gives you all the right incentives (and opportunities to find out if you're wrong). Post error sheets where appropriate to point out errors.
- Always be prepared to return back and check a data weirdness. There's surely at least one error in your data set that you haven't found yet. (Do your empirical work accordingly!)

Programming

- Always work with a pen/pencil and writing pad beside you; you'll probably use it more than you might imagine.
- Programs should be written such that a random person can read and understand it. Think some random person reading key output (which should be posted), or, more importantly … yourself.
- Each program should start off with a description of the name of the program, what language/package it's written for, what the program does and how it fits into the overall project.
- Comments should occupy about a quarter of the program. Write comments so that the program is self-explanatory at least to those who know the language. It encourages good code-writing. It also enables others interested in replication to … replicate. The most important person who should be able to figure out what's going on is you, in a few months/years.
- One idea deserves one program.
- More detail is better than less. For instance, URLs should be inserted wherever possible (don't say "available on the internet" but write "I downloaded this on 6-5-2011 from http://ddp-ext.worldbank.org/ext/DDPQQ/report.do?method=showReport")
- Write your programs so that changes to earlier stuff (e.g., a data set) is easy to handle (i.e., it should be easy to re-run a program if you find a mistake in an upstream input).
- Don't write more than three lines of code without figuring out some way to check your progress … and continually be aware of your ability to make subtle (and obvious!) mistakes.

- Naming conventions for programs; I use Wittgenstein's for the *Tractatus*. P1 is the first program. P1a is a perturbation of p1 that changes it somewhat; p1b might be a different perturbation. P1a1 is a perturbation of p1a. p2 does something quite different from p1.

- Naming conventions for data: start with data1. Do some stuff and write it out to a data set named data1. Then take that, add some more or change it in data2, and write it out to a data set named data2, etc. Keeping names of programs and output aligned is helpful; so is ordering programs in some accessible fashion. If you have different types of data sets, give them different names, e.g., csdatax.* and tsdatax.*

- Use "meta-programs" to allow for easy regeneration of your data sets and output.

    1. These operate by re-running programs sequentially. So a program "data" might regenerate all your data sets by running, in sequence, data1, data2, …. That's another reason to use clear naming conventions.

    2. Meta-programs are always a good idea; they make it easy to regenerate everything after you find a mistake. Lowering the cost of regenerating stuff makes you more likely to search hard for mistakes; you want to be as open as possible to such mistakes.

STATA

- Never read or a save a STATA data set without "d" and "sum" ing in – descriptive statistics quickly enable you to compare data sets. Always provide descriptive statistics right before you do any serious empirical work.

- "Notes" should be embedded within STATA data sets.

Posting

- Wherever possible, post your data set. There are a number of reasons:

    1. It's good science; replication should be easy.

    2. It's in your own self interest, for a number of reasons:

        a. It encourages you to be careful while you're creating your data.

        b. It allows you to find mistakes more easily (if/when others find them for you; it's reasonable to state that anyone who uses your data has an obligation to tell you if they find a problem).

c. It generates citations (it's reasonable to ask someone to cite the fact that they're using your data).

d. It's good for your reputation; you want to be known as someone whose empirics can be easily challenged, checked, and replicated.

- Sometimes data is confidential or proprietary. In that case, post output, and offer to run a comparable program for anyone who wants to check something. The data stays confidential, but you're still allowing others to challenge or check your work.

- I post only the final version of the data set; that's what people are typically most interested in. Of course you must be clear in your paper about the origins of the data series, and how you've massaged your data.

- I post output for most (and certainly key/representative) results. That way you can check your tables easily, and it also allows others to replicate your work easily, which provides a big incentive for you to be careful.