STATA Resources, Unit 4, Section 5.2
Closing comments on STATA in isolation
Kyle Puetz • 4/2/13

Goals for the class — An advanced STATA user should:

- 1) Be able to write, document, and implement a workflow data management
    - Includes:
        - Recoding variables
        - Merging and appending datasets
        - Reshaping data
        - Creating composite scores via a variety of techniques
- 2) Be able to write elegant do-files that use loops and programs to automate the tasks within STATA.
- 3) Make data management fully replicable so only the original data entry happens outside of STATA but all other data management is internal.
- 4) Be able to automate tasks in STATA through interface with other programs.
- 5) Develop programming skills so that you can achieve the same conceptual goal many ways.
- 6) Develop strong skills in commenting, documenting, and code-testing.
- 7) Integrate him- or herself into a larger community of STATA users.

<u>There are three themes in this lecture:</u>

1. The importance of master do-files in maintaining replicability:

As your programming becomes more complex, you'll find yourself using more and more do-files. For any given project, you may actually have numerous do-files that go into creating the dataset you ultimately analyze. A *master do-file*, whose only job is to activate all those do-files in the order in which they should run, is an important best practice. If your do-file only runs because you had some data in memory but said data would not necessarily be in memory at a later time and date, then you have not really created a replicable program. Run your program from a "clean STATA start"; if it doesn't work, you've got a problem.

Even when you know all the individual programs work that comprise a master do-file work on their own, you need to initiate them from a clean STATA start from a master do-file and see if that works. When that is true, you have fully replicable data.

2. Comment, comment, comment

When you're writing code, it seems really fresh in your mind. But you're going to send that paper off, it's going to be under review for three to *n* number of months, and you're going to get it back some period of time later, and you may have to alter your models. And when you return to your program, it's *not* going to be fresh in your mind anymore. Furthermore, when you reach a certain point in your career, you'll be working on several projects simultaneously, and even what you worked on last week may no longer be fresh on your mind. The importance and value of comments is that they permanently inscribe your intent into your programs. Be clear, be concise, and don't put it off till later.

3. Interpretability, parsimony, accuracy in programming

You can program things in multiple different ways, so you might be able to reasonably ask the question: Which of multiple different ways should I prefer? There are a couple different criteria you can consider in deciding how to program.

- 1) Which is going to be easiest for you to read later? Reading your code, in addition to your comments, is how you're going to be able to remember how you performed some task. Don't use as many abbreviations when you write statements; it takes so much more time to interpret the statement when you return to it. Make sure you're commenting as you go. Make certain there are notes at the beginning of the file about dependencies.

- 2) Elegance is important: Use loops when you can. Use programs when you can. Any time that codes becomes more compact, it is going to be easier to debug and easier to see the train of thought that led you to this end. Use something that has the fewest working parts. The more you type, the more errors you make, so type less and make your code do more.

- 3) Don't unnecessarily introduce errors into your work: Occasionally you need to be tricky or clever, but see if there's a straightforward way to do something. Generating a program that has been previously tested obviates the need to generate similar loops. The reduced likelihood of errors generated by using tested programs over loops is really marked.