The Stata Journal (2009)
9, Number 3, p. 504

Stata tip 79: Optional arguments to options

Nicholas J. Cox
Department of Geography
Durham University
Durham, UK
n.j.cox@durham.ac.uk

Programmers occasionally would like an option for a program to come in two flavors: a simple or default option, with no arguments, and a more complicated but more flexible alternative, with arguments. For example, the simple option might call up a graph with programmer-chosen defaults, while the complicated option might pass graph options to the graph command in question, signaling variations from those defaults.

With a simple trick, you can implement two options that appear to the user to be this single option that is either simple or complicated. Following age-old programmer jargon, let us imagine an option that can be `foobar` or `foobar(`*arguments*`)`.

Step 1: Declare to `syntax` that there are two options, say, `foobar` and `FOOBAR2()`, and the latter is precisely, say, `FOOBAR2(string)`. The outburst of uppercase letters indicates to `syntax` that the latter can be abbreviated `foobar()`. You can also indicate names that can be abbreviated more, say, `FOObar` and `FOObar2(string)`.

Step 2: Process input within your program. For example,

```
if "`foobar'`foobar2'" != ""
```

is a test of whether either option has been called. If `foobar2()` has been called, then the local macro `foobar2` will be defined and can be treated further. If the argument to `foobar2()` might itself contain quotation marks, then compound double quotes, `‘" "’`, are in order.

Step 3: In documentation for the user, you need not mention the two options but may merely declare that the syntax is that an argument is optional, e.g., `foobar`[`(`*string*`)`].

If curious users find out by looking at the code that the option with arguments is really `foobar2()`, no harm is done. They would be partway to working out, independently of this tip, how the optional options are coded. Browsing code and borrowing tricks that you want to use yourself remains one of the best ways to grow as a programmer.

