The Stata Journal (2008)
8, Number 3, pp. 446–447

Stata tip 65: Beware the backstabbing backslash

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

The backslash character, \, has two main roles for Stata users who have Microsoft Windows as their operating system. This tip warns you to keep these roles distinct, because Stata's interpretation of what you want may puzzle you. The problem is signaled at [U] **18.3.11 Constructing Windows filenames using macros** but nevertheless bites often enough that another warning may be helpful.

The first and better known role is that the backslash acts as a separator in full specifications of directory or filenames. Thus, on many Windows machines, a Stata executable may be found within `C:\Program Files\Stata10`. Note the two backslashes in this example.

The second and lesser known role is that the backslash is a so-called escape character that suppresses the default interpretation of a character.

The best example in Stata is that the left quotation mark, `, is used to delimit the start of local macro names. Thus `` `frog' `` to Stata is a reference to a local macro called `frog`. Typically, Stata sees such a reference and substitutes the contents of the local macro `frog` at the same place. If no such macro is visible, that is not a bug. Instead, Stata substitutes an empty string for the macro name.

What happens if you want the conventional interpretation of the left quotation mark? You use a backslash to flag that you want to override the usual Stata interpretation.

```
. display "He said \`frog´."
He said `frog´.
```

It is perhaps unlikely, although clearly not impossible, that you do want this in practice. The problem is that you may appear to Stata to specify this, even though you are likely to do it purely by accident.

Suppose, for example, that you are looping over a series of datasets, reading each one into Stata, doing some work, and then moving on to the next.

Your code may look something like this:

```
. foreach f in a b c {
.     use "c:\data\this project\`f´"
.     and so on
. }
```

Do you see the difficulty now? Your intent is that the loop uses a local macro, which in turn takes the values `a`, `b`, and `c` to read in the datasets `a.dta`, `b.dta`, and `c.dta`. But Stata sees the last backslash as an instruction to escape the usual interpretation of the

left quotation mark character that immediately follows. The resulting misunderstanding will crash your code.

Any Unix (including Macintosh) users reading this will feel smug, because they use the forward slash, `/`, within directory or filenames, and this problem never bites them. The way around the problem is by knowing that Stata will let you do that too, even under Windows. Stata takes it upon itself to translate between you and the operating system. You certainly need to use the forward slash in this example to escape the difficulty of the implied escape. Typing `use "c:\data\this project/‘f’"` would solve the problem. A backslash is only problematic whenever it can be interpreted as an escape character. In fact, you can mix forward and backward slashes willy-nilly in directory or filenames within Stata for Windows, so long as you do not use a backslash just before a left quotation mark.

The tidiest solution is to use forward slashes consistently, as in `use "c:/data/this project/‘f’"`, although admittedly this may clash strongly with your long-practiced Windows habits.