Stata tip 98: Counting substrings within strings

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

Consider the following problem, based on a real one reported on Statalist. A user has string date–times in differing forms; let us imagine they are in a variable, sdate. The desire is for everything to be \texttt{DMY hms}—that is, day, month, year, space, hours, minutes, and seconds. Examples of data are

\begin{quote}
"25/12/2010 11:22"
"25/12/2010 11:22:33"
"25/12/2010 11:22:33:444"
\end{quote}

Compared with the standard, the first example lacks seconds; the second example is fine; and the third example includes milliseconds, but expressed in a nonstandard way. Stata expects seconds and milliseconds in the decimal form “33.444”.

Some cleanup is required. One way forward is to note that the number of colons (:) present within the string is diagnostic of whether action is required and what to do. So how do you count substrings (in this case, just a colon) within strings? The problem has been aired in this journal at least once before (Cox 2011), but it occurs sufficiently often that it deserves a further flag. The main solution tends to provoke the reaction “Yes, of course!”, whether because people know it already or they now see that it is obvious and direct. However, people (including myself) have often supposed that the problem requires a more complicated approach than it really does.

Faced with this kind of problem, an experienced user tends to browse the help for string functions to see if there is a function dedicated to this problem, but in this case browsing will be in vain. But as so often happens, combining different functions is more successful. Consider how you would count for yourself. You would naturally work from one end of the string to the other, noting each occurrence. It is immaterial whether you count from left to right or from right to left, but note that Stata, by default, works on strings from left to right.

Stata has a function, \texttt{subinstr()}, that looks for occurrences of substrings within strings and replaces them with a specified substring (often just an empty string, \texttt{""}). This function gives us a solution. Consider the calculation

\begin{quote}
\texttt{length(sdate) - length(subinstr(sdate, ",", \"\", .))}
\end{quote}

As with elementary algebra, working from the inside of complicated expressions outward is a good tactic for understanding. The function call

\begin{quote}
\texttt{subinstr(sdate, ",", \"\", .)}
\end{quote}
N. J. Cox

...
That case is clearly more complicated than the case of counting occurrences disjointly. Fortunately, it appears to arise much less frequently.

Second, remember to adjust for the length of substring when counting substrings using the `subinstr()` method. `length("banana") - length(subinstr("banana", "ana", ",", .))` yields 3 because three characters were removed, but just one occurrence of the substring was. In general, divide by `length("substring")`. If you know what it is, there is a small efficiency gain in saying so. For example, you should not write

```stata
gen n_ana = (length(svar) - length(subinstr(svar, "ana", ",", .))) / length("ana")
```

because that obliges Stata to calculate `length("ana")` for every observation. Even in a programming situation where the particular substring is not predictable in advance, a calculation like

```stata
local sslen = length("substring")
gen n_substring = (length(svar) - length(subinstr(svar, "substring", ",", .))) / `sslen`
```

allows the result to be used in macro form in a command:

```stata
gen n_substring = (length(svar) - length(subinstr(svar, "substring", ",", .))) / `sslen`
```

That way, the contents of the macro are substituted before the `generate` command gets to work so that it uses a known constant rather than an expression to be evaluated for every observation.

The problem could get more complicated, yet. For example, the substring concerned might vary from observation to observation, as when conventions about reporting are variable in the data. So long as the substring is known and included within a variable (say, `ssvar`), this is no more difficult than any previous problem.

```stata
gen sscount = (length(svar) - length(subinstr(svar, ssvar, ",", .))) / length(ssvar)
```

Alternatively, we might be counting different possible types of substring, for which we just need to cycle over all the possibilities.

**Reference**