Published in: The Stata journal vol. 18 (2018), no. 1
[Postprint]

Stata tip 129: Efficiently processing textual data with Stata's new Unicode features

Alexander Koplenig
Department of Lexical Studies
Institute for the German language (IDS)
Mannheim, Germany
koplenig@ids-mannheim.de

Prior to Stata 13 and especially Stata 14, Stata's abilities to process natural language data were limited because of the string length limit of 244 characters and the lack of Unicode support. To extract basic descriptive information from unformatted text data (for example, word frequency information), one needed to rely on workarounds such as Benoit's (2003) `wordscores` implementation. With Stata 14, this situation has changed. To demonstrate why the new string-processing capabilities of Stata are highly relevant and useful for anyone who deals with natural language data, let us consider, for example, that we want to extract the five most frequent words of the English Universal Declaration of Human Rights. We can do this by first downloading the text file from http://www.unicode.org/udhr/ using the `copy` command. Then, we can use the new string functions `ustrwordcount()` and `ustrword()` to produce language-specific Unicode words that are based on word-boundary rules or dictionaries for languages that do not use spaces between words (for example, for Thai, see below):

```
. copy "http://unicode.org/udhr/d/udhr_eng.txt" udhr.raw, replace

. clear

. local words=ustrwordcount(fileread("udhr.raw"))

. set obs `words´
number of observations (_N) was 0, now 1,963

. generate word=ustrword(fileread("udhr.raw"),_n)

. contract word

. gsort -_freq

. list in 1/5

     +--------------+
     | word   _freq |
     |--------------|
  1. |  the     120 |
  2. |  and     106 |
  3. |    ,      95 |
  4. |   of      93 |
  5. |   to      83 |
     +--------------+
```

Note that Stata automatically separates punctuation tokens from actual word tokens. In many situations, this is convenient because it makes (effortful) cleaning procedures unnecessary.

In a similar vein, it is easy to extract frequency statistics for n-grams that are sequences of n-consecutive word tokens. Let us say we want to extract the five most frequent word pairs (that is, 2-grams) from the data above. We can do this by generating a word identifier that records the position of each word in the text. The resulting file consisting of all first words of each 2-gram is temporarily stored and then merged with all second words:

```
. generate long wordidentifier=_n

. rename word word1

. tempfile TEMP

. save `TEMP´, replace
(note: file F:\ST_0j000002.tmp not found)
file F:\ST_0j000002.tmp saved

. replace wordidentifier=wordidentifier-1
(1,963 real changes made)

. rename word1 word2

. merge 1:1 wordidentifier using `TEMP´, keep(3)
    Result                           # of obs.
    -----------------------------------------
    not matched                             0
    matched                             1,962  (_merge==3)
    -----------------------------------------

. contract word1 word2

. gsort -_freq

. order word1 word2 _freq

. list in 1/5

     +--------------------------+
     | word1     word2    _freq |
     |--------------------------|
  1. |     .   Article       30 |
  2. | right        to       28 |
  3. |   the     right       28 |
  4. |   has       the       25 |
  5. |    of       the       23 |
     +--------------------------+
```

Note that another possibility to extract the most frequent n-grams and corresponding (absolute or relative) frequencies would be to use the `groups` command written by Cox (2017), instead of using the `contract` command before sorting and listing.

As written above, for languages that do not use spaces between words, using the functions `ustrwordcount()` and `ustrword()` has the additional advantage that Stata takes care of the word segmentation. For example, if we want to extract the five most frequent words of the Thai Universal Declaration of Human Rights, we just download the Thai text file. Interestingly, we do not have to change the *loc* argument in the `ustrwordcount()` and `ustrword()` functions. This is because Stata uses the ICU tokenizer, which automatically switches to dictionary-based rules when it identifies particular Unicode script input.

```
. copy "http://unicode.org/udhr/d/udhr_tha.txt" udhr.raw, replace

. clear

. local words=ustrwordcount(fileread("udhr.raw"))

. set obs `words'
number of observations (_N) was 0, now 2,385

. generate word=ustrword(fileread("udhr.raw"),_n)

. contract word

. gsort -_freq

. list in 1/5

     +--------------+
     | word   _freq |
     |--------------|
  1. |  และ     107 |
  2. |  การ      91 |
  3. |   จะ      75 |
  4. |  ที่       72 |
  5. |   ใน      71 |
     +--------------+
```

While the ICU documentation (http://userguide.icu-project.org/boundaryanalysis) lists dictionary-based support for Japanese, Khmer, Chinese, and Thai, one can find that other languages, such as Burmese (language-specific ISO 639-3 code: `mya`), Lao (`lao`), or Tibetan (`bod`), are also supported by using the corresponding ISO code in the `copy` command above (`udhr_ISO.txt`).

1 References

Benoit, K. 2003. Wordscores: Software for coding political texts. http://www.tcd.ie/Political_Science/wordscores/software.html.

Cox, N. J. 2017. Speaking Stata: Tables as lists: The groups command. *Stata Journal* 17: 760–773.