Community Learning Resource: Regular Expressions

Andrew Davis
Feb. 24th, 2016
SOC 561

A previous blogpost by Nadina: http://soc596.blogspot.com/ explored some great examples of how to use these commands in Stata. This post will take things a bit further with new examples and exercises. I'll begin with a review of the basics of "regular expressions" and then move to the new examples and exercises.

Using "Regular Expressions" in Stata

What is a "regular expression" in general?

-A regular expression is a sequence of characters that define a search pattern.
    *Commonly used in word in terms of "find and replace" function. Many people have probably used a function such as this in a word document. But…
-Word is not alone. Many programs allow for similar capacities, including Stata.

Literal expressions v. regular expressions (Theory of Regular Expressions)
-When to use what?
    *Regular expressions are generally too powerful if you just want to seek out a single concept. They can open you to risk of error when a simple "find and replace" function might have been best.
    *Regular expressions are generally good for searching out multiple variables with a similar concept (factors), and patterns within data. This can be very helpful in organizing and manipulating your data.

Stata
-You use search techniques to find values of variables in a dataset that is brought into Stata.
    *You can only use this for "string" variables.
        - String variables have words as values in Stata, as opposed to numbers

Are my variables "string?"
-You can easily find out if your variables, or which variables are "string" using the "describe" command in stata.

Below is an example of the use of a "describe" command, as well as output on 8 variables with different storage types in this dataset. In this case, the variable "country" is stored as a "string" variable.

```
. describe

Contains data from C:\Users\APD\Desktop\Sociology of Conflict Data Sets\EPR3Country Wimmer.
  obs:         7,908                          epr v3.01 country level data (31 Dec 2014)
 vars:            80                          31 Dec 2014 11:16
 size:     1,747,668
-------------------------------------------------------------------------------------------
              storage   display    value
variable name   type    format     label      variable label
-------------------------------------------------------------------------------------------
yearc           long    %10.0g                Year-country
year            int     %ty                   Year
cowcode         int     %10.0g                Country code Correlates of War
country         str32   %32s                  State name
gdpcap          float   %9.0g                 GDP per capita, mostly PWT 7.1 and WDI 2012
gdpcapl         float   %9.0g                 GDP per capita, lagged
oilpc           float   %9.0g                 Oil production per capita, various sources
oilpcl          float   %9.0g                 Oil production per capita, lagged
```

Regular expressions in Stata continued…

Above and beyond simple “find and replace” type functions, Stata allows the user to seek out patterns in the data using simple commands and symbols (see table below).

Counting	
*****	Asterisk means “match zero or more” of the preceding expression.
+	Plus sign means “match one or more” of the preceding expression.
?	Question mark means “match either zero or one” of the preceding expression.
Characters	
a–z	The dash operator means “match a range of characters or numbers”. The “a” and “z” are merely an example. It could also be 0–9, 5–8, F–M, etc.
.	Period means “match any character”.
****	A backslash is used as an escape character to match characters that would otherwise be interpreted as a regular-expression operator.
Anchors	
^	When placed at the beginning of a regular expression, the caret means “match expression at beginning of string”. This character can be thought of as an “anchor” character since it does not directly match a character, only the location of the match.
$	When the dollar sign is placed at the end of a regular expression, it means “match expression at end of string”. This is the other anchor character.
Groups	
|	The pipe character signifies a logical “or” that is often used in character sets (see square brackets below).
[ ]	Square brackets denote a set of allowable characters/expressions to use in matching, such as [a-zA-Z0-9] for all alphanumeric characters.
( )	Parentheses must match and denote a subexpression group.

Source: http://www.stata.com/support/faqs/data-management/regular-expressions/

Using the Commands

Each command (regexm, regexr and regexs) indicate to Stata that you would like to use a (re)gular (ex)pression.

Regexm: you want Stata to find a match (m). (Is there a phone number?)

First, and most basic is the command "regexm." As reviewed under "theory of regular expressions" regexm should be used to find a pattern within some data.

The syntax for regexm:

gen *newvar* = regexm (*stringvar*, expression)

-The key components of this expression are the function (the regexm command) and the expression (what you're asking the function to search for).
-This is pretty intuitive, regexm searches for whatever you are looking for within the variable.

Regexr: you want Stata to replace (r) the expression. ("Let's replace those phone numbers")

-You should use "regexr" when you want to replace a portion of a string variable.

The syntax for regexr:

gen/replace *newvar* = regexr (*stringvar*, "expression", "*replace"*)

-The key components of this syntax are the "regexr" which commands Stata to replace a portion of a string.
-This "portion of a string" can be found in the parentheses. *Stringvar* locates the variable of which you want to replace a portion, "expression" refers to what you'd like to replace in the variable, and the final "replace" refers to what you would like to put in the place.

Regexs: you want Stata to isolate a subsection (s) of a larger string. ("Let's see those phone numbers, pull 'take' them out and put them into a new variable")
-Like with the above functions, you should only use this expression in the service of seeking out a bona-fide pattern in your data.
-To use this expression, you must use syntax that combines regexm and regexs. In general you want to create a new variable that *is* the isolate of the string.

The syntax for regexs:

gen *newvar* =regexs(#) if regexm(*stringvar*, ("first subexpression") ("second subexpression")...("nth subexpression"))

-There are several important components of this expression.

* First, the # sign as highlighted above represents the portion of the string you would like to isolate. For instance, if your phone number was (520)867-5309 you would use regexs (0) to return the entire phone number, regexs (1) to return (520), regexs (2) to return 867, and regexs (3) to return 5309….(I got it!).

Subexpression #	String returned
0	1march2014
1	1
2	march
3	2014

*Second, the end of the syntax "regexm(*stringvar*, ("first subexpression") ("second subexpression")…("nth subexpression"))" should be handled carefully, depending on what you are wanting to return. Please refer carefully to the list of symbols used in regular expressions in Stata before moving forward.

~See the forthcoming example for a good demonstration on using this command.

Examples:

The following will serve as a guide, I will apply each expression type to an example using Stata syntax. I will be working with the publicly available "Minorities at Risk" dataset, linked here: http://www.cidcm.umd.edu/mar/mar_data.asp#quantitativemar

1. **Using regexm:**

The variable we will be working with is vmar_region

Cross-tabulation appears as follows:

```
. tab vmar_region
```

VMAR_Region	Freq.	Percent	Cum.
Asia	174	20.42	20.42
Latin America and the Caribbean	99	11.62	32.04
Middle East and North Africa	87	10.21	42.25
Post-communist States	180	21.13	63.38
Sub-Saharan Africa	225	26.41	89.79
Western Democracies and Japan	87	10.21	100.00
Total	852	100.00	

-We will use regexm to create a variable that combines all responses from Africa (excluding North Africa and the Middle East).

The syntax is as follows:

gen africa=regexm(vmar_region, "Sub")

```
. tab africa
```

africa	Freq.	Percent	Cum.
0	627	73.59	73.59
1	225	26.41	100.00
Total	852	100.00	

-You can also use regexm to produce lists, below is an example from the Minorities at Risk dataset:

```
list country if regexm(country, "Republic") == 1
```

	country
22.	Dominican Republic
23.	Dominican Republic
24.	Dominican Republic
166.	Czech Republic
167.	Czech Republic
168.	Czech Republic
169.	Czech Republic
170.	Czech Republic
171.	Czech Republic

2. Using regexr:

The variable we will be working with in this example (in the Minorities at Risk dataset) is “autonend” which is a measure of the year in which a minority group lost political autonomy. A snapshot of cross tabulation of this variable looks like this.

```
. tab autonend
```

AUTONEND	Freq.	Percent	Cum.
-99	333	39.22	39.22
0	18	2.12	41.34
1077	3	0.35	41.70
1410	3	0.35	42.05
1447	3	0.35	42.40
1521	6	0.71	43.11
1524	3	0.35	43.46
1528	3	0.35	43.82
1532	3	0.35	44.17
1534	6	0.71	44.88
1537	3	0.35	45.23
1538	6	0.71	45.94
1550	3	0.35	46.29
1552	3	0.35	46.64
1567	3	0.35	47.00
1576	3	0.35	47.35
1580	3	0.35	47.70
1599	3	0.35	48.06
15th century	3	0.35	48.41

…with numbers reaching until the modern day.

-We want to replace all values of lost autonomy that occurred in the 1500's or before with a value called: preenlightenment. This is how you'd go about doing that:

Syntax:

gen preenlightenment=regexr(autonend, "[0-1][0-5][0-9][0-9]", "preenlightenment")

I will now tabulate the variable "preenlightenment" which should reflect the replace change that was made to these values. A snapshot of this variable is presented below, reflecting the change.

1975	6	0.71	88.34
1976	3	0.35	88.69
1979	3	0.35	89.05
1989	6	0.71	89.75
1990	6	0.71	90.46
1994	3	0.35	90.81
1998	3	0.35	91.17
1999	3	0.35	91.52
19th century	3	0.35	91.87
early 20th century	6	0.71	92.58
mid 1800s	6	0.71	93.29
preenlightenment	57	6.71	100.00
Total	849	100.00	

-As you can see from this snapshot, all values from the 1500's and before are collapsed into the "preenlightenment" value.

3. Using regexs:

We will continue to use the "autonend" variable to demonstrate how to effectively use regexs.

-As previously discussed, regexs isolates a portion of a string variable. In this case we would like to isolate cases that occurred in the 1500s and before or have text associated with them, such as the value "15th century".

Syntax:

gen century = regexs(1) if regexm(autonend, "([0-9][0-5][0-9][0-9])[\-]*[0-9]*[a-zA-Z]*$")

-A tabulation of this output is produced below. As you can see, values have been isolated from the string variable "autonend" into a variable to which I've added the label "16th century measures."

```
. tab century
```

16th century measures	Freq.	Percent	Cum.
1077	3	5.26	5.26
1410	3	5.26	10.53
1447	3	5.26	15.79
1521	6	10.53	26.32
1524	3	5.26	31.58
1528	3	5.26	36.84
1532	3	5.26	42.11
1534	6	10.53	52.63
1537	3	5.26	57.89
1538	6	10.53	68.42
1550	3	5.26	73.68
1552	3	5.26	78.95
1567	3	5.26	84.21
1576	3	5.26	89.47
1580	3	5.26	94.74
1599	3	5.26	100.00
Total	57	100.00	

```
.
```

Exercises:

Like much programming, using regular expressions take quite a bit of practice to get the hang of. To get started here are four exercises that will get you on your way to understanding how to use regular expressions in Stata.

Each of these exercises use data from the "1978 autos" dataset. To retrieve these data, simply type "sysuse auto" into your Stata browser.

1. The goal of this exercise is to get you familiar with the regexm command. Please open the "1978 autos" and complete the following tasks.
    a. List which variables you can use regular expressions on.
    b. List vehicles whose make is "Toyota"
    c. List vehicles whose make is "Pont."
    d. List vehicles with the letters "VW" in their make
2. Using the "1978 autos" dataset, use regexs to create a variable that includes only cars with numbers in their name. Provide evidence that your variable does what you want it to do.
3. Using the "1978 autos" dataset, use regexs to create a variable that includes only VW's. Provide evidence that your variable does what you want it to do.
4. Use regexr to replace VW with Volkswagon. Provide evidence that your variable does what you want it to do.