Using regular expressions for data management in Stata

Rose Anne Medeiros
rosem@ats.ucla.edu

Statistical Consulting Group
Academic Technology Services
University of California, Los Angeles

2007 West Coast Stata Users Group meeting

Outline

What are regular expressions?

- A relatively easy, flexible method of searching strings. You can use them to search any string (e.g. variables, macros).
- In Stata, there are three *functions* that use regular expressions.
- Regular expressions are *not* the solution to every problem involving strings. In most cases the built in string functions in Stata will do at least as good a job, with less effort, and a lower probability of error.

What are regular expressions?

- A relatively easy, flexible method of searching strings. You can use them to search any string (e.g. variables, macros).
- In Stata, there are three *functions* that use regular expressions.
- Regular expressions are *not* the solution to every problem involving strings. In most cases the built in string functions in Stata will do at least as good a job, with less effort, and a lower probability of error.

- **regexm(*s*,*re*)** allows you to search for the string described in your regular expressions. It evaluates to 1 if the string matches the expression.
- **regexs(*n*)** returns the n*th* substring within an expression matched by regexm (hence, regexm must always be run before regexs).
- **regexr(*s1*,*re*,*s2*)** searches for *re* within the string (*s1*) and replaces the matching portion with a new string (*s2*).

- **regexm(*s*,*re*)** allows you to search for the string described in your regular expressions. It evaluates to 1 if the string matches the expression.
- **regexs(*n*)** returns the n*th* substring within an expression matched by regexm (hence, regexm must always be run before regexs).
- **regexr(*s1*,*re*,*s2*)** searches for *re* within the string (*s1*) and replaces the matching portion with a new string (*s2*).

- **regexm(*s*,*re*)** allows you to search for the string described in your regular expressions. It evaluates to 1 if the string matches the expression.
- **regexs(*n*)** returns the n*th* substring within an expression matched by regexm (hence, regexm must always be run before regexs).
- **regexr(*s1*,*re*,*s2*)** searches for *re* within the string (*s1*) and replaces the matching portion with a new string (*s2*).

What do regular expressions look like?

- In Stata they are always enclosed in quotation marks.
- They can include both strings you wish to match exactly, and more flexible descriptions of what to look for.
  - Strings typed directly are matched exactly (literals), e.g. "a" only matches "a".
  - Operators are characters that appear in square brackets (i.e. [and]), they are matched more flexibly, or are other characters that describe how they should be matched.
    - `. * + ? ^ $ | ( ) [ ] \`
    - Values inside brackets may include ranges, e.g. 0-9, a-z, A-Z, f-x, 0-3.
- For example if we wanted to find the area codes in a list of phone numbers we could use:

```
"^[\(]?[0-9][0-9][0-9]"
```

What do regular expressions look like?

- In Stata they are always enclosed in quotation marks.
- They can include both strings you wish to match exactly, and more flexible descriptions of what to look for.
  - Strings typed directly are matched exactly (literals), e.g. "a" only matches "a".
  - Operators are characters that appear in square brackets (i.e. [and]), they are matched more flexibly, or are other characters that describe how they should be matched.
    - `. * + ? ^ $ | ( ) [ ] \`
    - Values inside brackets may include ranges, e.g. 0-9, a-z, A-Z, f-x, 0-3.
- For example if we wanted to find the area codes in a list of phone numbers we could use:

```
"^[\(]?[0-9][0-9][0-9]"
```

What do regular expressions look like?

- In Stata they are always enclosed in quotation marks.
- They can include both strings you wish to match exactly, and more flexible descriptions of what to look for.
  - Strings typed directly are matched exactly (literals), e.g. "a" only matches "a".
  - Operators are characters that appear in square brackets (i.e. [and]), they are matched more flexibly, or are other characters that describe how they should be matched.
    - . * + ? ^ $ | ( ) [ ] \
    - Values inside brackets may include ranges, e.g. 0-9, a-z, A-Z, f-x, 0-3.
- For example if we wanted to find the area codes in a list of phone numbers we could use:

```
"^[\(]?[0-9][0-9][0-9]"
```

What do regular expressions look like?

- In Stata they are always enclosed in quotation marks.
- They can include both strings you wish to match exactly, and more flexible descriptions of what to look for.
  - Strings typed directly are matched exactly (literals), e.g. "a" only matches "a".
  - Operators are characters that appear in square brackets (i.e. [and]), they are matched more flexibly, or are other characters that describe how they should be matched.
    - `. * + ? ^ $ | ( ) [ ] \`
    - Values inside brackets may include ranges, e.g. 0-9, a-z, A-Z, f-x, 0-3.
- For example if we wanted to find the area codes in a list of phone numbers we could use:

```
"^[\(]?[0-9][0-9][0-9]"
```

What do regular expressions look like?

- In Stata they are always enclosed in quotation marks.
- They can include both strings you wish to match exactly, and more flexible descriptions of what to look for.
  - Strings typed directly are matched exactly (literals), e.g. "a" only matches "a".
  - Operators are characters that appear in square brackets (i.e. [and]), they are matched more flexibly, or are other characters that describe how they should be matched.
    - `. * + ? ^ $ | ( ) [ ] \`
    - Values inside brackets may include ranges, e.g. 0-9, a-z, A-Z, f-x, 0-3.
- For example if we wanted to find the area codes in a list of phone numbers we could use:

```
"^[\(]?[0-9][0-9][0-9]"
```

Example 1: Extracting zip codes

- We have a list of addresses stored in a string variable, and we want to extract the zip codes.
- What do we want to search for?
  - A five-digit number (`[0-9][0-9][0-9][0-9][0-9]`)
- Are there any complications? (Of course there are!)
  - Some addresses include zip+4
  - Some addresses include the country
  - Some addresses have five-digit street numbers

Example 1: Extracting zip codes

- We have a list of addresses stored in a string variable, and we want to extract the zip codes.
- What do we want to search for?
  - A five-digit number (`[0-9][0-9][0-9][0-9][0-9]`)
- Are there any complications? (Of course there are!)
  - Some addresses include zip+4
  - Some addresses include the country
  - Some addresses have five-digit street numbers

Example 1: Extracting zip codes

- We have a list of addresses stored in a string variable, and we want to extract the zip codes.
- What do we want to search for?
  - A five-digit number (`[0-9][0-9][0-9][0-9][0-9]`)
- Are there any complications? (Of course there are!)
  - Some addresses include zip+4
  - Some addresses include the country
  - Some addresses have five-digit street numbers

The data

address
4905 Lakeway Drive, College Station, Texas 77845 USA
673 Jasmine Street, Los Angeles, CA 90024
2376 First street, San Diego, CA 90126
66666 West Central St, Tempe AZ 80068
12345 Main St. Cambridge, MA 01238-1234
12345 Main St Sommerville MA 01239-2345 usa
12345 Main St, Watertown MA 01233 USA

```
"([0-9][0-9][0-9][0-9][0-9])[\-]*[0-9]*[ a-zA-Z]*$"
```

Presto!

address	zip
4905 Lakeway Drive, College Station, Texas 77845 USA	77845
673 Jasmine Street, Los Angeles, CA 90024	90024
2376 F street, San Diego, CA 90126	90126
66666 West Central St, Tempe AZ 80068	80068
12345 Main St. Cambridge, MA 01238-1234	01238
12345 Main St Sommerville MA 01239-2345 usa	01239
12345 Main St, Watertown MA 01233 USA	01233

How'd she do that?

```
gen zip = regexs(1) if /*
  */ regexm(address, "([0-9][0-9][0-9][0-9][0-9])[\-]*[0-9]*[ a-zA-Z]*$")
list
```

Example 2: Cleaning Data

- In an online survey respondents were asked the number of days in the last month they engaged in some activity.
- Some respondents entered just a number, as desired.
- Other respondents entered other values.
- -999 was used to represent a missing value.

The data

days
never
0
20+
-999
0
35
0
Never
8 plus
every day
15 days
8
-999
3
12-14

What needs to be done:

- Change "never" and "Never" with 0.
- Change "every day" to 30 (or 31).
- Remove the word "days" where it appears.
- Remove "+" and "plus."
- Replace -999 with a missing value.
- Change illegal values (e.g. 35) to some other value.

Presto!

days	days2
never	0
0	0
20+	20
-999	.
0	0
35	.
0	0
Never	0
8+	8
every day	30
15 days	15
8	8
-999	.
3	3
12-14	12

How'd she do that?

```
* Create a variable that will equal 0 if there is a legal numeric
* value (0-31) and nothing else for the variable days, and 1 otherwise.
gen flag1 = 1
replace flag1=0 if regexm(days, "(^[0-9]$)|(^[1-2][0-9]$)|(^30$)|(^31$)")

* -999 is a missing value, so these don't need to be flagged either.
replace flag1=0 if(days=="-999")

* generate a new variable to contain the cleaned (numeric only) values.
gen days2 = .

* If days contains a legal numeric value, set days2 = days
replace days2 = real(days) if(flag1==0&days!="-999")

* List the values that days takes on when it is not a numeric value.
list days if flag1==1

* replace "never" or "zero" with zero
replace days2 = 0 if(regexm(days, "[Nn]ever|[Zz]ero"))
```

```
* For cases containing "days" or "times" look for numbers
* a valid number at the start of a line
replace days2 = real(regexs(1)) /*
        */ if(regexm(days, "(^[0-9]+)[ ]*(times|days)"))

* If the respondent reported a range of numeric values,
* return only the first.
replace days2 = real(regexs(1)) if(regexm(days, "([0-9]+)(\-[0-9]+)"))
replace days2 = real(regexs(1)) /*
        */ if(regexm(days, "([0-9]+)[ ]*to[ ]*([0-9]+)"))

* replace +, plus, and or more with reported value
replace days2 = real(regexs(1)) /*
        */ if(regexm(days, "([0-9]+)[ ]*(\+|plus|or more)"))

* Replace "every day" with 30
replace days2 = 30 if(regexm(days, "[eE]very[ ]*[dD]ay[.]*"))

* Check to make sure all values of days2 are believable,
* and filled in as much as possible
list id days days2 if(days2<0|days2>31|days2==.)
```

Where can I go from here?

- The official Stata FAQ on regular expressions: http://www.stata.com/support/faqs/data/regex.html
- UCLA's Academic Technology Services' page on regular expressions: http://www.ats.ucla.edu/stat/stata/faq/regex.htm