Use Perl Regular Expressions in SAS® Shuguang Zhang, WRDS, Philadelphia, PA

ABSTRACT

Regular Expression (Regexp) enhance search and replace operations on text. In SAS®, the INDEX, SCAN and SUBSTR functions along with concatenation (||) can be used for simple search and replace operations on static text. These functions lack flexibility and make searching dynamic text difficult, and involve more function calls. Regexp combines most, if not all, of these steps into one expression. This makes code less error prone, easier to maintain, clearer, and can improve performance.

This paper will discuss three ways to use Perl Regular Expression in SAS: 1. Use SAS PRX functions; 2. Use Perl Regular Expression with filename statement through a PIPE such as 'Filename fileref PIPE 'Perl programm'; 3. Use an X command such as 'X Perl_program';

Three typical uses of regular expressions will also be discussed and example(s) will be presented for each: 1. Test for a pattern of characters within a string; 2. Replace text; 3. Extract a substring.

INTRODUCTION

Perl is short for "Practical Extraction and Report Language". Larry Wall Created Perl in mid-1980s when he was trying to produce some reports from a Usenet-Nes-like hierarchy of files. Perl tries to fill the gap between low-level programming and high-level programming and it is easy, nearly unlimited, and fast.

A *regular expression*, often called a *pattern* in Perl, is a template that either matches or does not match a given string. That is, there are an infinite number of possible text strings. A given pattern divides that infinite set into two groups: the ones that match, and the ones that do not. There is never any kinda-sorta-almost-up-to-here wishywashy matching: the ones that match, and the ones that do not. A pattern may match just one possible string, or just two or three, or a dozen, or a hundred, or an infinite number or it may match all strings *except* for one, or except for some, or except for an infinite number

Perl regular expressions were added to SAS in Version 9. SAS regular expressions (similar to Perl regular expressions but using a different syntax to indicate text patterns) have actually been around since version 6.12, but many SAS users are unfamiliar with either SAS or Perl regular expressions. Both SAS regular expressions (the RX functions) and Perl regular expressions (the PRX functions) allow you to locate patterns in text strings.

Regular Expression (Regexp) enhance search and replace operations on text. The INDEX, SCAN and SUBSTR functions along with concatenation (||) can be used for simple search and replace operations on static text. These functions lack flexibility and make searching dynamic text difficult, and involve more function calls. Regexp combines most, if not all, of these steps into one expression. Regular expressions can provide a much more compact solution to a complicated string manipulation task. Regular expressions are especially useful for reading highly unstructured data streams. This makes code less error prone, easier to maintain, clearer, and can improve performance.

Regexp can be used to:

- Test for a pattern of characters within a string. For example, a string can be examined to determine if it contains a correctly formatted telephone number. This is called data validation.
- Replace text. Regexp can be used to find specific text within a string. The found text can be removed or replaced with other text.
- Extract a substring from a string. Regexp can be used to find and easily extract text found within a string.

There are other ways to use Perl regular expression in SAS including using Perl regular expression with filename statement through PIPE such as 'Filename fileref PIPE 'Perl programm' or using an X command to run a Perl program from within SAS such as 'X Perl_program'.

This paper will first discuss three ways to use Perl Regexp in SAS, and then will briefly discuss three major applications of Perl Regexp in SAS.

THREE WAYS TO USE PERL REGEXP IN SAS

In this section, three ways to use Perl regular expression in SAS in Windows environment and UNIX environment will be introduced.

1. USE SAS PRX FUNCTIONS

Perl regular expressions were added to SAS in Version 9. Names of all the PRX functions and CALL routines in Base SAS begin with the prefix PRX:

PRXparse

- regex-id = prxparse(Perl-regex)
- Compile Perl regular expression Perl-regex and return regex-id to be used by other PRX functions.

PRXmatch

- pos = prxmatch(regex-id | Perl-regex, source)
- Search in **source** and return position of match or zero if no match is found.

PRXchange

- new-string = prxchange(regex-id | Perl-regex, times, old-string)
- Search and replace times number of times in oldstring and return modified string in new-string

Call PRXchange

- call prxchange(regex-id, times, old-string, newstring, res-length, trunc-value, num-of-changes)
- Same as prior example and place length of result in *res-length*, if result is too long to fit into *new-string*,
 trunc-value is set to 1, and the number of changes is placed in *num-of-changes*

Call PRXsubstr

- CALL PRXSUBSTR(regexp-id, source, pos, <length>)
- Return position and length of where a regexp matches in source

Call PRXnext

- CALL PRXNEXT(regexp-id, start, stop, source, pos, length)
- Return position and length of where a regexp matches in source starting at start and ending at stop

PRXposn

- text = prxposn(regex-id, n, source)
- After a call to prxmatch or prxchange, prxposn return the text of capture buffer n.

Call PRXposn

- call prxposn(regex-id, n, pos, len)
- After a call to prxmatch or prxchange, call prxposn sets pos and len to the position and length of capture buffer n.

Call PRXnext

- call prxnext(regex-id, start, stop, source, pos, len)
- Search in source between positions start and stop. Set pos and len to the position and length of the match. Also set start to pos+len+1 so another search can easily begin where this one left off.

Call PRXdebug

- call prxdebug(on-off)
- Pass 1 to enable debug output to the SAS Log.

Pass 0 to disable debug output to the SAS Log.

Call PRXfree

- call prxfree(regex-id)
- Free memory for a regex-id returned by **prxparse**.

Example: Using PRX functions in data step

Program:

```
DATA _NULL_;

IF _N_ = 1 THEN PATTERN = PRXPARSE("/batman/");

RETAIN PATTERN;

INPUT STRING $40.;

POSITION = PRXMATCH(PATTERN, STRING);

FILE PRINT;

PUT PATTERN= STRING= POSITION=;

DATALINES;

attention: batman in this line

spiderman in this line
;
```

Output:

```
PATTERN=1 STRING=attention: batman in this line POSITION=12
PATTERN=1 STRING=spiderman in this line POSITION=0
```

In this example, the PRXPARSE compiles Perl regular expression and return regex-id "PATTERN" to be used by PRX function – "PRXMATCH"; "PRXMATCH" searches in input strings and return position of match or zero if no match is found.

2. USE PERL REGULAR EXPRESSION WITH FILENAME STATEMENT THROUGH A PIPE

A pipe is a channel of communication between two processes. Pipes enable your SAS application to receive input from any UNIX/DOS/Perl command that writes to standard output and to route output to any UNIX/DOS/Perl command that reads from standard input. In UNIX/DOS commands, the pipe is represented by a vertical bar (|).

In order to use Perl regular expression in Windows environment, Perl needs to be installed in Windows. One of the free Perl software is Active PERL which can be downloaded at: http://www.activestate.com/downloads/.

You can use the FILENAME statement to assign filerefs not only to external files and I/O devices, but also to a pipe. The syntax of the FILENAME statement is

- FILENAME fileref PIPE 'UNIX/DOS/Perl-command' <options>;
- fileref is the name by which you reference the pipe from SAS.
- PIPE identifies the device-type as a UNIX/DOS pipe.
- 'UNIX/DOS/Perl-command' is the name of a UNIX/DOS/Perl command, executable program, or shell script to which you want to route output or from which you want to read input. The command(s) must be enclosed in either double or single quotation marks.
- Options control how the external file is processed.

When the fileref is used for reading, the specified UNIX/DOS/Perl command executes, and any output sent to its standard output or standard error is read through the fileref.

Example: Sending the Output of the Process Command to a SAS DATA Step in UNIX

The following SAS program uses the PIPE device-type keyword to send the output of the UNIX/Perl command to a SAS DATA step. The resulting SAS data set contains data about every process currently running SAS:

Input File:

```
Batman and Robin
Robin and Spiderman
```

Program:

Output:

```
Batman and Robin
Robin and Batman
```

The operating environment uses pipes (|) to send the output from cat command to the Perl regular expression command (substitution), which replace every occurrence of the word "Spiderman" with "Batman". The FILE-NAME statement connects the output of the Perl "substitute" command to the fileref "inf". The DATA step then creates a data set named test from the INFILE statement that points to the input source. The INPUT statement reads the first 20 characters on each input line.

Whether you are using the command as input or output depends on whether you use the fileref in a reading or writing operation. For example, if the fileref is used in an INFILE statement, then SAS assumes that the input comes from a UNIX command; if the fileref is used in a FILE statement, then SAS assumes that the output goes to a UNIX command.

Example: Sending the Output of the Process Command to a SAS DATA Step in Windows

The following SAS program in Windows uses the PIPE device-type keyword to send the output of the DOS/Perl command to a SAS DATA step. The resulting SAS data set contains data about every process currently running SAS:

Input File:

```
Batman and Robin
Robin and Spiderman
```

Program:

Output:

```
Batman and Robin
Robin and Batman
```

This example is Windows version of example 1. The DOS command "type" is used to replace "cat" command in UNIX and the Perl "substitute" command does not need to be quoted Windows environment as in UNIX environment.

3. USE X COMMAND

As mentioned above, you can either use PRX functions in data step or use Perl with filename statement through a PIPE to take the advantage of the power of Perl regular expression to clean up the data before save them to permanent SAS datasets. You can also call Perl program in SAS session to preprocess the raw data before reading into SAS.

You can execute Perl program (or Windows or MS-DOS or UNIX commands) from within SAS by using the X command. The X command can be run outside of a DATA step. You can enter the X command in the command bar or any SAS command line.

The X command has the following syntax:

X <'command'>:

This form of the X command issues one command. The command is passed to the operating environment and executed. If errors occur, the appropriate error messages are displayed.

Example: Execute Perl program from within SAS to clean up raw data in Windows/UNIX

Original Raw Data - batman.txt

```
Batman and Robin
Robin and Spiderman
```

Perl program – clean_data.pl (Windows version)

Perl program - clean data.pl (UNIX version)

```
# clean_data.pl - UNIX Version
                                       #
                                       #
# clean up data for SAS program
open (INFILE,"/home/NESUG/examples/batman.txt")
    or die "can't open the input file";
open (OUTFILE,">/home/NESUG/examples/batman_new.txt")
   or die "can't open output file";
select (OUTFILE);
while (<INFILE>){
 s/Spiderman/Batman/g;
 print;
close (INFILE);
close (OUTFILE);
```

Data after clean-up - batman_new.txt

```
Batman and Robin
Robin and Batman
```

SAS program (Windows Version):

```
/*____*/
/* program name : example_x.sas - Windows Version */
/* description : this program first using X command */
/* to execute a Perl program to clean up the raw */
/* data and then read the clean data into SAS */
/*----*/
/*----*/
/* execute Perl program clean_data.pl to clean up */
/* and store the cleaned data into a new file
/*-----
x 'Perl C:\NESUG\examples\clean data.pl';
/* read the cleaned data - batman_new.txt into SAS */
data test;
infile 'C:\NESUG\examples\batman_new.txt' truncover;
length string $ 20;
input string $ 1-20;
run;
proc print;
run;
```

SAS program (UNIX Version):

```
/*_____*/
/* program name : example_x.sas - UNIX Version */
/* description : this program first using X command */
/* to execute a Perl program to clean up the raw */
   data and then read the clean data into SAS
/*----*/
/* execute Perl program clean_data.pl to clean up */
/* and store the cleaned data into a new file
/*----*/
x 'Perl /home/NESUG/examples/clean_data.pl';
/*----*/
/* read the cleaned data - batman_new.txt into SAS */
/*----*/
infile '/home/NESUG/examples/batman new.txt' truncover;
length string $ 20;
input string $ 1-20;
run;
proc print;
run;
```

Output:

Batman and Robin Robin and Batman

In this example, when we run the SAS program example_x.sas, the Perl program - clean_data.pl is executed from within SAS to read the original data (batman.txt) into Perl, substitute every occurrence of the word "Spiderman" with "Batman", and then write the cleaned data into a new file (batman_new.txt). SAS then read the cleaned data (batman_new.txt) into SAS dataset test.

TYPICAL USES OF PERL REGULAR EXPRESSIONS

There are three typical uses of Perl regular expression in SAS:

- · Test for a pattern of characters within a string
- Replace text
- Extract a substring

SAS has a very detailed documentation on the above usages of Perl regular expression. Please visit the following SAS support page for further information:

http://support.sas.com/rnd/base/topics/datastep/Perl_regexp/regexp.motivation.html

CONCLUSIONS

Perl regular expression can be used in SAS in three ways to enhance search and replace operations on text. It is simple, flexible and very powerful. Regular expressions can provide a much more compact solution to a complicated string manipulation task. It makes code less error prone, easier to maintain, clearer, and can improve performance.

REFERENCES

Schwartz, R.L., and Phoenix, T. (2001), Learning Perl, Third Edition, O'Reily & Associates.

Wall, L., Christiansen, T., and Orwant, J. (2000), Programming Perl, Third Edition, O'Reily & Associates.

An Introduction to Perl Regular Expressions (SUGI 31)

http://www2.sas.com/proceedings/sugi31/110-31.pdf

Perl Regular Expressions Tip Sheet from SAS:

http://support.sas.com/rnd/base/topics/datastep/Perl_regexp/regexp-tip-sheet.pdf

PRX Function Reference from SAS:

http://support.sas.com/rnd/base/topics/datastep/Perl_regexp/regexp2.html

Using Regular Expressions from SAS

http://support.sas.com/rnd/base/topics/datastep/Perl_regexp/regexp.motivation.html

ACKNOWLEDGMENTS

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are trademarks of their respective companies.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact:

Shuguang Zhang Wharton Research Data Service (WRDS) The Wharton School of University of Pennsylvania 3733 Spruce Street, Vance Hall 216 Philadelphia, PA 19104 Sgzhang8@wharton.upenn.edu