SAS® GLOBAL FORUM 2016

IMAGINE. CREATE. INNOVATE.

How to Use PERL Functions in SAS®: Some Basic Examples

Peter Timusk
Statistics Canada
Centre for Special Business Projects
Production Officer

#SASGF

How to Use PERL Functions in SAS®: Some Basic Examples

Peter Timusk

Statistics Canada, Centre For Special Business Projects

ABSTRACT

PERL is a good language to work with text and in the case of SAS® with character variables. PERL was much used in website programming in the early days of the World Wide Web. SAS provides some PERL functions in DATA step statements. PERL is useful for finding text patterns or simply strings of text and returning the strings or positions in a character variable of the string. I often use PERL to locate a string of text in a character variable. Knowing this location, I know where to start a SUBSTR function. The poster covers a number of PERL functions available in SAS® 9.1 and later. Each function is explained with written text, and then a brief code demonstration shows how I use the function. The examples are jointly explained based on SAS documentation and my best practices.

PERL FUNCTIONS

- PRXMATCH FUNCTION (See the slide here)
- Searches for a pattern match and returns the position at which the pattern is found.
- PRXPARSE FUNCTION (See the slide here)

Compiles a Perl regular expression (PRX) that can be used for pattern matching of a character value.

- CALL PRXNEXT ROUTINE (See the slide here)

Returns the position and length of a substring that matches a pattern, and iterates over multiple matches within one string.

- Other PERL functions.

The author has not used all the PERL functions available in SAS and is less familiar with these other functions and call routines . There are also these functions

- PRXCHANGE
- PRXPARSEN
- PRXPOSN
- And there are a variety of PERL call routines that work in combination with the functions. The SAS online documentation makes these all very accessible to leant to use.

How to Use PERL Functions in SAS®: Some Basic Examples

Peter Timusk

Statistics Canada, Centre For Special Business Projects

PRXMATCH

- PRXMATCH: A PERL function used often is PRXMATCH which takes two parameters, the instructions of what text you are looking for and the location to search. Here is the SAS 9.2 documentation on the syntax of this to be exact.

PRXMATCH (regular-expression-id | perl-regular-expression, source)

This is what will work for regular expression at the simplest form of regular expression.

PRXMATCH ('/ROW_/', UPCASE(Domain))

This will find location of the text 'Row_ ' in the value of the character variable Domain. It will return an integer that is the position in the character variable Domain of the location of the letter R in the expression 'Row_'. There are in the Domain variable a number of labels for where a statistic in the same record goes in an excel workbook. Row_00, Row_01, Row_02 etc. correspond to Excel workbook rows such as this.

USE OF PRXMATCH

Region (Label in published Table)	Row (Label inside calculation process. This is not published.)	Statistic from Calculation
Atlantic	Row_00	30%
Quebec	Row_01	34%
Ontario	Row_02	31%

Variable Domain	PRXMATCH
Stat='RES' Row_02='02'	12
Stat='RES' dom_1='1' Row_02='02'	22

RowKey= SUBSTR (Domain, PRXMATCH ('/ROW_/',UPCASE(Domain))+8,2)

RowKey='02'

Here we have found the location of the value of the Row_02 8 characters after the R in Row_02, if the Row string follows a regular pattern where the value is always 8 characters after the R in 'Row_' then this use of PRXMATCH allows one to tell SUBSTR where to start the text capture. The location of the string 'Row_' in the variable Domain varies depending on if there is also a dom_1 label present or not.

How to Use PERL Functions in SAS®: Some Basic Examples

Peter Timusk

Statistics Canada, Centre For Special Business Projects

PRXPARSE

PRXPARSE: Allows one to write PERL regular expressions to define a text string one is searching for. From SAS documentation this is stated as

Compiles a Perl regular expression (PRX) that can be used for pattern matching of a character value.

Here is the SAS syntax documentation for PRXPARSE.

regular-expression-id=PRXPARSE (perl-regular-expression)

Here are some basic examples from my work.

```
ExpressionID1 = prxparse('/[R][0-9]{6}/');

ExpressionID2 = prxparse('/[col_][0-9]{2}/');
```

USE OF PRXPARSE

ExpressionID1

Again we see that this is surrounded by a matching pair of '/ and /' open and closing characters. The square brackets define what we are looking for in Expression ID1 we are looking for an R then any number 0-9 and the curly brackets tell us how many of the proceeding characters (in the square brackets proceeding) we are looking for. In the case of ExpressionID1 we are looking for text like this: the capital letter R followed by 6 numbers. Like this R123456 or this R937100 or this R931187. In our work there were names of variables that followed this convention.

ExpressionID2

Again we see that this is surrounded by a matching pair of '/ and /' open and closing characters. The square brackets define what we are looking for in Express ion ID2 we are looking for the letters 'col' and underscore exactly like this 'col_' then any number 0-9 and the curly brackets tell us how many of the proceeding characters we are looking for. In the case of ExpressionID2 we are looking for text like this col_ followed by 2 numbers. Like this col_01, col_04, or col_98. In our work these are names of variables that we that we code for Excel columns of statistics we publish in Excel tables.

How to Use PERL Functions in SAS®: Some Basic Examples

Peter Timusk

Statistics Canada, Centre For Special Business Projects

PRXNEXT

- PRXNEXT: A PERL call routine used for scanning a text source for repeated occurrences of a PERL regular expression.. Here is the SAS 9.2 documentation on the syntax of this to be exact.

```
call prxnext(ExpressionID1, start, stop, text , position, length);
```

This is what will work for regular expression at the simplest form of regular expression.

```
ExpressionID1 = prxparse('/[R][0-9]{6}/');

call prxnext(ExpressionID1, start, stop, text , position, length);
   do while (position > 0);

Newvariable="I"|| trim(left( substr(SASline, position+1, length-1)));
```

Here again, like the last example this searches for variables in the text that have the form R123456 and then takes this variable name and makes a new name that starts with the letter I but has the same numbers.

USE OF PRXNEXT

In fact, we are searching lines of SAS program code read into the variable SASline for these variables that begin with R and the R stands for response variable as in respondent to a survey. Then we are replacing the letter R with the letter I standing for imputation as in imputation variable. Another team has created a flag in these I variables indicating the original R variable has been imputed for this record.

This is allowing us to take a program written for the response variables and in fact some 400 programs and automatically write 400 new programs that use the corresponding imputation variables but in in a different way. In this case, we are computing an imputation rate in the second set of programs.

How to Use PERL Functions in SAS®: Some Basic Examples

Peter Timusk

Statistics Canada, Centre For Special Business Projects

RESULTS

PERL can be used dynamically to adjust to changing or unknown character variables values. Also when analyzing text such as a SAS program we can find text we need and then modify this text and reuse it in other ways. We have also not really explored the more complex regular expressions possible and the PERL regular expressions although perhaps complex to understand can be used with care to suit more complex situations. The author has written some complex PERL regular expressions by having a regular expressions guide side by side the code window and working slowly.

CONCLUSIONS

Even more uses can be found with use of more of the PERL functions available in SAS. Also the author has not fully explore the combination of PERL functions and PERL call routines. We would suggest starting simple and reading some of the PERL documentation for the PERL language, as well as, the SAS documentation for PERL in the references for this poster. You will be impressed with what PERL can manage to help you with in your SAS programming.

REFERENCES

–

Cody, Ron. May 9-12, 2004 “An Introduction to Perl Regular Expressions in SAS 9“ SUGI 29 Montréal, Canada SAS Institute. http://www2.sas.com/proceedings/sugi29/265-29.pdf (Online accessed March 15th, 2016) SAS Institute.

SAS Institute, “Perl Regular Expressions Tip Sheet” Https://support.sas.com/rnd/base/datastep/perl_regexp/regexp-tip-sheet.pdf (Online accessed March 15th, 2016) SAS Institute.

SAS Institute, “Pattern Matching Using Perl Regular Expressions (PRX)”, “SAS 9.4 “SAS(R) 9.4 Functions and CALL Routines: Reference, Fourth Edition“ http://support.sas.com/documentation/cdl/en/lefunctionsref/67960/HTML/default/viewer.htm#n13as9vjfj7aokn1syvfyrpaj7z5.htm (Online accessed March 15th, 2016) SAS Institute.

SAS® GLOBAL FORUM 2016

IMAGINE. CREATE. INNOVATE.

LAS VEGAS | APRIL 18-21

#SASGF