PharmaSUG 2019 – BP-315

Using the PRXCHANGE Function to Remove Dictionary Code Values from the Coded Text Terms

Lynn Mullins, PPD, Cincinnati, OH

ABSTRACT

SAS® Perl regular expression (PRX) functions and CALL routines refer to a group of functions and CALL routines that use a modified version of the Perl programming language as a pattern-matching language to parse character strings. Some of the tasks that you can perform using PRX functions are:

- Search for a pattern of characters within a string
- Extract a substring from a string
- Search and replace text with other text
- Parse large amounts of text, such as Web logs or other text data

You can write SAS® programs that do not use regular expressions to produce the same results as you do when you use Perl regular expression functions. However, the code without the regular expression functions requires more SAS® functions to handle character positions in a string and to manipulate parts of the string.

Perl regular expression functions combine most, if not all, of these steps into one SAS® statement. The resulting code is less prone to error, easier to maintain, and clearer to read. This paper will discuss how to use Pearl regular expression functions to remove those pesky codes that are sometimes at the end of dictionary coded text terms.

INTRODUCTION

In the pharmaceutical industry sometimes, programmers encounter dictionary coded terms in the raw data, such as, Adverse Event or Concomitant medication terms with the term code embedded in the text of the term. Many times, these dictionary codes are placed several null values or spaces after the actual text of the term or at the very end of the text field with the last part of the code being in the maximum length of the field. The embedded codes are not consistently placed in the same position of the field making it difficult to find them and according to CDISC standards, the code should not be included in the value of the term field in the final SDTM data set. Therefore, the programmer needs to remove these codes from the text term field prior to creating the data set.

One way to find and remove these codes is to use Perl regular expression functions. The use of PRX functions easily removes these unwanted codes found at the end of the text fields.

PERL REGULAR EXPRESSION FUNCTION BENEFITS

SAS® programs without the use of Perl regular expression functions can be used to produce the same results as Perl regular expression functions. However, code without the use of Perl regular expressions requires the use of more SAS® functions such as compress, substr, tranwrd, etc. to handle character positions in a string and to manipulate parts of the string. Using PRX functions combine most, if not all, of these steps into one SAS® statement. The resulting code is less prone to error, easier to maintain, and clearer to read.

Perl regular expression functions can be used in the DATA step to enhance search-and-replace options in text. Some of the tasks that can be performed using PRX functions are:

- validate data
- extract a substring from a string
- replace text

COMPONENTS OF A PERL REGULAR EXPRESSION

Perl regular expressions consist of characters and special characters that are called metacharacters. When performing a match, SAS searches a source string for a substring that matches the Perl regular expression that you specify. Using metacharacters enables SAS to perform special actions. These actions include forcing the match to begin in an identified location and matching a specified set of characters. Paired forward slashes are the default delimiters.

PERL METACHARACTERS AND CONSTRUCTS

BASIC PERL METACHARACTERS AND THEIR DESCRIPTIONS

Table 1 below describes some of the basic Perl metacharacters.

Metacharacter	Description
\A	matches a character only at the beginning of a string.
\b	matches a word boundary (the position between a word and a space): "er\b" matches the "er" in "never" "er\b" does not match the "er" in "verb."
\B	matches a non-word boundary: "er\B" matches the "er" in "verb" "er\B" does not match the "er" in "never".
\cA-\cZ	matches a control character. For example, \cX matches the control character control-X.
\C	matches a single byte.
\d	matches a digit character that is equivalent to [0−9].
\D	matches a non-digit character that is equivalent to [^0−9].
\l	specifies that the next character is lowercase.
\L	specifies that the next string of characters, up to the \E metacharacter, is lowercase.
\w	matches any word character or alphanumeric character, including the underscore.
\W	matches any non-word character or non-alphanumeric character and excludes the underscore.

Table 1 Basic Perl Metacharacters

PERL GENERAL CONSTRUCTS

Table 2 below describes some of the Perl general constructs.

Metacharacter	Description
()	Indicates grouping
non-metacharacter	matches a character
* + ? \	to match these characters, override (escape) with \

Table 2 Perl General Constructs

RAW DATA SET CODING VARIABLES

The text variable DRUG_NAME in the concomitant medication raw data will be used to show how to remove the unwanted dictionary code from the value using Perl regular expressions.

For this example, the DRUG_NAME field contains the value of part of the CODE variable embedded in slashes “/” and we want to remove the CODE value including the slashes while leaving the actual DRUG_NAME text as is.

The raw data in Table 3 contains the three records, two with term codes embedded in the term name text and one where the term name text contains a forward slash. Table 4 shows how the raw data needs to be split.

DRUG_NAME
PROVENTIL /00139501/
TYLENOL /00020001/
TYLENOL W/CODEINE

Table 3 Raw Data

DRUG_NAME	Unwanted characters	PREFERRED_CODE
TYLENOL	/00020001/	00020001005
PROVENTIL	/00139501/	00139501001
TYLENOL W/CODEINE		00116401196

Table 4 Raw Data Split

Please note that the TYLENOL W/CODEINE DRUG_NAME includes a forward slash "/" and the CODE value is missing.

PERL REGULAR EXPRESSIONS (PRX): PRXCHANGE FUNCTION

We are creating a new field named CMDECOD by using the PRXCHANGE function with a Perl regular expression against the original DRUG_NAME field.

The PRXCHANGE function is used to remove the code values embedded in slashes. The basic syntax is:

```
CMDECOD = strip(prxchange('s/(\D\d\d\d\d\d\d\d\d\D)/ /', -1, DRUG_NAME));
```

The arguments are:

- ( ) Specifies grouping
- s Specifies the search string
- \D Specifies to match a non-digit character (i.e. "/")
- \d Specifies to match a digit character
  - There are 9 "\d" – one for each number.
- / / Specifies the replacement string is a blank
- -1 Specifies that matching patterns continue to be replaced until the end of the source is reached
- DRUG_NAME is the source string to be searched

CHANGED DATA SET CODING VARIABLES AFTER USING PRXCHANGE

Table 5 shows the results of using the Perl Regular expression function, PRXCHANGE to remove the dictionary term code from the dictionary term name. The unwanted characters are removed in the new field CMCODE including the slashes. Note that the slash "/" in the DRUG_NAME field containing the text "TYLENOL W/CODEINE" was not removed.

DRUG_NAME	CODE
TYLENOL	00020001005
PROVENTIL	00139501001
TYLENOL W/CODEINE	00116401196

Table 5 Results from PRXCHANGE

CONCLUSION

Perl Regular expression functions can be used in SAS® statements to manipulate text data. It is an efficient and easy way to find text within one statement without having to use multiple SAS® functions. There are many Perl Regular expression functions and this paper showed how to use one, PRXCHANGE, to find dictionary term

REFERENCES

Regular-Expressions.Info. 2019. Accessed September 15, 2018. http://www.regular-expressions.info/index.html.

SAS®9 – Perl Regular Expressions Tip Sheet. Accessed September 15, 2018. http://support.sas.com/rnd/base/datastep/perl_regexp/regexp-tip-sheet.pdf.

Using Perl Regular Expressions in the DATA Step. Accessed September 15, 2018. http://support.sas.com/documentation/cdl/en/lefunctionsref/63354/HTML/default/viewer.htm#p1vz3ljudbd756n19502acxazevk.htm.

Campbell, Joes. "Perl Regular Expressions in SAS® 9.1+ - Practical Applications", Proceedings of the Pharmaceutical SAS Users Group 2012 Conference. Available at https://www.pharmasug.org/proceedings/2012/TA/PharmaSUG-2012-TA08.pdf.

ACKNOWLEDGMENTS

Thanks to PPD Management for their reviews and comments. Thanks to my family for their support.

DISCLAIMERS

The contents of this paper are the work of the authors and do not necessarily represent the opinions, recommendations, or practices of PPD.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Lynn Mullins
PPD
(910) 558-4343
Lynn.mullins@ppdi.com