Paper PO03

THE MEDCODE MACRO AND THE PRXCHANGE FUNCTION

Gary Stevens, Biogen Idec Ltd, Maidenhead, UK

ABSTRACT

SAS© PRX functions use a modified version of Perl regular expression for compilation and matching. This paper demonstrates a macro developed to search and replace superfluous information which appeared on coded concomitant medications following implementation of the 2007 version of WHODRUG, using the SAS prxchange function.

BACKGROUND

Following an update to a more recent version of the WHODRUG coding dictionary (the 2007 version), by a service provider contracted to perform the data management activities for one of our studies, we started to receive unexpected coded concomitant medication terms on the CM domain CDISC dataset (SDTM version 3.1.1). Some of the coded concomitant medications, or the CMDECOD variable, appeared with a series of numbers between two forward slashes appended in an ad-hoc position after the expected coded term. Further investigation confirmed that this extra information was a concatenation of the WHONUM and WHOSEQ1 variables and was added to indicate the method of action of the drug. Although more useful to physicians and investigators it was decided to remove this superfluous information for reporting purposes. After careful analysis of the problem, a decision was made to use the SAS PRXCHANGE function as it seemed the simplest way to search and replace this extra information with white space and therefore effectively remove the unwanted alphanumeric string from the coded drug name. The original coded term with the extra information is not removed from the output file as the modified coded term is simply added as an extra variable to the output file.

POSSIBLE SOLUTIONS

Originally many coding solutions were put forward to remove this superfluous information, mostly from programmers who had come across this issue in tables they were producing and had dealt with this within their table code. This task was more often than not completed in numerous lines of code using a whole host of INDEX, INDEXC, SCAN and SUBSTR functions. All of these methods were validated for correctness; while we did find instances where code worked 99% of the time, there were one or two instances where CMDECOD (Coding variable) values did not fit the pattern used in the programmer's code.

EXAMPLE OF A POSSIBLE SOLUTION

This is one possible coding solution.

```
%macro medcode(indata=,outdata=, byvar=, invar= ,newvar=);

      data &outdata(drop=pos1);
      set &indata;
            by &byvar;
                  pos1=index(&invar,'/');
                  if pos1>0 then do;
                        if substr(&invar,(pos1+9),1)='/' then do;
                              &newvar=trim(left(substr(&invar,1,pos1-1)));
                        end;
                        else &newvar=&invar;
                  end;
                  else &newvar=&invar;
      run;

%mend medcode;

%medcode(cm1,cm2,usubjid,cmdecod,newdecod);
```

This 12 line piece of code is fairly simple to follow and will produce the desired result, but again, only in nearly all cases. There are certain instances where this could produce undesired truncation of the CMDECOD variable. If there are 2 forward slashes

within the medication name separated by 8 characters the CMDECOD will be truncated at the point where the first forward slash occurs. Likewise a forward slash in the CMDECOD string and again at the start of the extraneous information beginning 9 characters later would also truncate the medication name in an undesired manner.

What we were looking for was a simple search and replace piece of code that could be incorporated into a macro, would search for a 'numeric' string of 8 characters within forward slashes, and create a new variable as an exact replica of the CMDECOD variable but with this numeric string and forward slashes removed. One suggestion put forward was to look at the SAS PRXCHANGE function, a new function included in SAS 9, which uses a modified version of Perl regular expression and matching.

ABOUT PERL

Perl (an acronym of Practical Extraction and Reporting Language) is a high level, general purpose, interpreted, dynamic programming language which was originally developed in 1987 by Larry Wall who was working as a systems administrator for NASA. More than 22 years of development of Perl has transformed it into an extremely powerful programming tool. It has a free distribution policy and is totally supported and further developed by its users. As of 2009 Larry Wall is still involved in the development of Perl and the next major release will be Perl 6, currently known as Topaz . However it is expected to be several years before Topaz achieves enough robustness, compatibility, portability, and performance to replace Perl 5 for ordinary use.

SAS PRX functions use a modified version of Perl 5.6.1 to perform regular expression compilation and matching. Only Perl regular expressions are accessible from the PRX functions. The PRXCHANGE function searches the variable source with the regular-expression-id. It returns the value in source with the changes that are specified by the regular expression. If there is no match the PRXCHANGE function returns the unchanged value in source.

THE MEDCODE MACRO – SIMPLIFIED

This is exactly what we needed and the whole macro below is just 5 lines with the search and replace part reduced to just one single statement.

```
%macro medcode(indata=, outdata=, byvar=, invar=, newvar=);

       data &outdata;
       set &indata;
              by &byvar;
                     &newvar = prxchange('s/\/\d*\// /',-1,&invar);
       run;

%mend medcode;
```

SYNTAX OF THE PRXCHANGE FUNCTION

The Basic syntax for the PRXCHANGE function is:

output variable name=prxchange(<Perl regular expression> , <pattern replacement> , source);

This is the syntax of the prxchange function as it is used in the macro.

```
NEWDECOD = prxchange('s/\/\d*\// /',-1,CMDECOD);
```

The Perl regular expression is encased within single quotes

```
's/\/\d*\// /'
```

The Perl regular expression is better to understand once broken down into components:

1. The s/ is the substitution operator.
2. The first part of the unwanted string is a '/' therefore a '\' is used as a delimiter both before and after the '/'.
3. The d means after the '/' there is any digit 0-9.
4. The * means 0 or more times.
5. '\' is the next delimiter and ends the numeric portion of the string to replace.
6. '/' is present as the last character of string to search.
7. There is a space between two forward slashes '/' and this is the replacement text i.e. white space.

The -1 means that the function should replace all matching patterns, and the source is the variable (original coded term) with the possible appendage of the unwanted alphanumeric string.

An alternative use of the Perl regular expression is show below. In this case the d* part of the regular expression can be replaced with a d repeated the number of times the numeric portion of the string is in length. Although slightly longer this will produce the same result.

```
NEWDECOD =prxchange('s/\/\d\d\d\d\d\d\d\d\// /',-1,CMDECOD)
```

RESULTS

Here is a short selection of results. Notice that CMDECOD values with a slash appearing within the medication name are not affected by the macro.

CMDECOD	NEWDECOD
MULTIVITAMIN /00831701/	MULTIVITAMIN
IBUPROFEN	IBUPROFEN
ZOLPIDEM TARTRATE	ZOLPIDEM TARTRATE
IBANDRONATE SODIUM	IBANDRONATE SODIUM
TYLOX /00446701/	TYLOX
PHYSIOTHERAPY	PHYSIOTHERAPY
WOBENZYM N /01199401/	WOBENZYM N
MULTIVITAMINS W/MINERALS	MULTIVITAMINS W/MINERALS
TOLTERODINE L-TARTRATE	TOLTERODINE L-TARTRATE
EUGYNON /00022701/	EUGYNON
BACITRACIN W/POLYMYXIN	BACITRACIN W/POLYMYXIN

CONCLUSIONS

The medcode macro has been fully validated by the Biogen Idec Macro Review Team and is present in their Global macro library. The prxchange function was a new concept to most of the team as it only became available in SAS 9.0. The original code discussed earlier in the paper is not 100% foolproof or as efficient as the 100% effective code using the prxchange function.

ACKNOWLEDGMENTS

Thank you to Sri Manivannan – SRG, Research Triangle Park , NC USA

CONTACT INFORMATION

Gary Stevens
Biogen Idec Ltd.
Innovation House
70 Norden Road
Maidenhead
Berkshire SL6 4AY
UK

Tel. +44 (0) 1628 501000
Direct +44 (0) 1628 512583
Fax +44 (0) 1628 501010

gary.stevens@biogenidec.com