PharmaSUG 2013 - Paper CC35

A One Line Method to Extract a Substring from a String using PRX

Joel Campbell, Advanced Analytics, Wilmington, NC

ABSTRACT

Perl Regular Expressions (PRX) are powerful tools available in SAS® and many other programming languages and utilities which allow precise and flexible pattern matching. The SAS 9 Language Reference provides examples to extract a substring from a string using PRX functions that require, at minimum, 5 elements and are potentially difficult to follow for someone looking for a basic example. In this paper, I'll demonstrate a single line method to extract a substring from a string using only the PRXCHANGE function.

INTRODUCTION

Extracting a substring from a string is a basic programming task – and one that any SAS user might need to perform. Prior to PRX, SAS functions like SUBSTR and INDEX were commonly used in conjunction to accomplish the task and worked well for simple applications. However, if a minimum amount of complexity is added to the task (for example, if the 2nd occurrence of a substring in a string is targeted) then the basic combination of INDEX and SUBSTR likely morphs into a *nested* combination of INDEXes and SUBSTRs. Add any more complexity to the target substring (for example, if case-insensitivity is desired when searching for the target substring) and the resulting code likely needs an extra function or two and/or wraps to several lines to fit on the screen.

Fortunately for SAS users, a family of Perl Regular Expression functions became available in SAS 9 which makes extracting a substring from a string much easier and gives the user access to the power of Perl regular expression functionality.

Unfortunately for SAS users attempting to learn the basics of using PRX to extract a substring from a string, the examples found at support.sas.com are unnecessarily complex and do not work well when it comes to providing simple examples. Don't get me wrong, I love support.sas.com and use it regularly with success… I just think that the PRX examples could be improved.

EXTRACTING A SUBSTRING FROM A STRING (SUPPORT.SAS.COM EXAMPLE)

Traditional implementation of PRX in SAS to extract a substring from a string involves using the PRXPOSN function, but this approach requires a three-statement-combination of PRX functions to accomplish the task, and is usually accompanied by a RETAIN statement and an IF _N_=1 THEN DO block, which brings the tally to 5 elements needed to extract a substring from a string using PRX as provided in the example from support.sas.com below (source: http://support.sas.com/documentation/cdl/en/lrdict/64316/HTML/default/viewer.htm#a002291852.htm).

```
data _null_;
   length first last phone $ 16;
   retain re;
   if _N_ = 1 then do;
      re=prxparse("/\(([2-9]\d\d)\) ?[2-9]\d\d-\d\d\d\d/");
   end;

   input first last phone & 16.;
   if prxmatch(re, phone) then do;
      area_code = prxposn(re, 1, phone);
      if area_code ^in ("828" "336" "704" "910" "919" "252") then
         putlog "NOTE: Not in North Carolina: "
                       first last phone;
    end;
   datalines;
Thomas Archer    (919)319-1677
Lucy Mallory       (800)899-2164
Tom Joad              (508)852-2146
Laurie Jorgensen  (252)352-7583
;
run;
```

The five elements required in this example are bolded and colored in red. It's not until the PRXPOSN function is used that the substring is actually being extracted from the string. A programmer may feel overwhelmed by the amount of code seemingly required to extract a substring from a string using PRX, and might pass up PRX altogether in favor of the more familiar INDEX/SUBSTR approach.

THE PRXCHANGE FUNCTION

In addition to the three PRX functions of PRXPARSE, PRXMATCH, and PRXPOSN required for the example above from support.sas.com an additional PRX function, PRXCHANGE, is available for the main purpose of performing find-and-replace type tasks within a string. The basic syntax is shown in the simple example below where all the occurrences of "find" would be replaced with "replace."

```
ReturnedString = PRXCHANGE( 's/find/replace/' , -1 , InputString );
```

Note first, that the "s" at the beginning of the regular expression indicates that a substitution will be performed. Secondly, the second parameter of "-1" indicates that all occurrences will be found and replaced.

THE CAPTURE BUFFER

In addition to the simple example just provided, it is possible to save portions of a matched string to the capture buffer for use in the replacement using the "(" and ")" with PRXCHANGE. In this example, the string "Goodnight, Jim" is replaced by "Jim Goodnight." If encountered, the string "Goodnight" is saved to capture buffer 1, while "Jim" is saved to capture buffer 2, these can be recalled in the replace string using $1 and $2. The following example would change "Goodnight, Jim" to "Jim Goodnight."

```
NAME = prxchange('s/(Goodnight), (Jim)/$2 $1/i',-1,NAME);
```

It is important to note that the PRXCHNGE function can save-to and recall-from the capture buffer in a single function call, whereas with the PRXPOSN function can only recall-from the capture buffer; when using PRXPOSN the PRXMATCH function is used to first save-to the capture buffer.

MATCHING A PORTION VERSUS MATCHING THE WHOLE

It is important to understand the difference between matching only a portion of the searched expression versus matching the whole expression when a target string is found. For example, there may be times that you want to change only portions of text found within a larger string or there may be times you want to modify the entire string if a certain target string is found within. In these examples, both regular expressions return true if the target string is found.

```
if prxmatch("/Target/i",myvar); ** Only matches target string **;
```

returns true for the same cases as:

```
if prxmatch("/^.*Target.*$/i",myvar); ** Matches the entire value of MYVAR **;
```

In this example, the "^" and "$" match the beginning and end of the searched string, respectively, and the two instances of ".*" allow for any combination of characters, including no characters at all. So the regular expression might be read as: starts with anything, contains "Target," and ends with anything… which is only slightly, yet importantly different from: contains "Target."

EXTRACTING A SUBSTRING FROM A STRING (SINGLE-LINE METHOD EXAMPLE)

To perform the single-line method of extracting a substring from a string using PRX, it is important to match the whole string. To simplify the approach: we're going to search a string for the substring, and then change the entire string to the substring we're interested in extracting. In this example I'll use the original perl regular expression from the support.sas.com example except for the important additions of "^.*" to the beginning and ".*$" to the end of the regular expression so that we'll be matching the whole string when the target expression is found.

```
data _null_;
   length first last phone $ 16;
   input  first last phone & 16.;

   area_code=prxchange("s/^.*\(([2-9]\d\d)\) ?[2-9]\d\d-\d\d\d\d.*$/$1/",-1,phone);

   if area_code ^in ("828" "336" "704" "910" "919" "252") then
         putlog "NOTE: Not in North Carolina: "
                        first last phone;
   datalines;
Thomas Archer    (919)319-1677
Lucy Mallory       (800)899-2164
Tom Joad             (508)852-2146
Laurie Jorgensen  (252)352-7583
;
run;
```

In this approach the regular expression matches the whole string and changes it to only the value of interest, which is then passed to the variable being set. In this way, a single line of code may be used to extract a substring from a string using a PRX function in order to set a variable.

CONCLUSION

The PRX functions, available beginning in SAS version 9.1, are powerful tools which allow for precise and flexible pattern matching. While traditional implementations of PRX to extract a substring from a string require 5 elements and even more lines of code, it is possible to extract a substring from a string with the power of PRX in a single line of code using the PRXCHANGE function.

CONTACT INFORMATION

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

Name: Joel Campbell
Enterprise: Advanced Analytics, Inc.
Address: P.O. Box 966
City, State ZIP: Wilmington, NC 28402
E-mail: Joel.Campbell@AdvancedAnalyticsCRO.com
Web: www.AdvancedAnalyticsCRO.com