Perl functions in SAS: Perl functions can add pearl in your code

Kamlesh Patel
Rang Technologies Inc, Piscataway, New Jersey
Agenda

- Introduction
- Fundamentals
  - Using character string in slashes
  - Using text strings in PRX functions
  - Making PERL regular expressions
- PRX Functions for beginners
- Examples to start with
Introduction: Basics

- Character variable in SAS
  - Various variables in clinical programming
  - Data manipulations
  - Many functions like INDEX, INDEXW, LENGTH, SUBSTR, SCAN, TRNWRD, FIND etc.

- Can we do anything more than this?
  - PRX Functions
Introduction: PRX Functions

- Based on PERL 5.6.1 (which is powerful character string manipulation language)
- PERL functions in SAS DATA Step
  - Since SAS v9.0

- Various functions and call routines are available based on PERL
- E.g. PRXMATCH
Introduction: PRX Functions

- **Can perform** -
- **String search**: Search for a specific string in character value
- **Extract out substring**: To take out a specific substring
- **Search + Replace**: Replace specific string in place of another string
- **Parse string**: Parse large amounts of text like a website or any other text data
FUNDAMENTALS AND BASICS OF PRX

1. USING CHARACTER STRING IN SLASHES

- PERL language - use slash for the string.
- Hence, any string constant should be written as –
  /text string/
- If text string, - Hospital,
  /Hospital/
- In SAS, character value – “quoted”,
- hence,
  ‘/Hospital/’
2. USING TEXT STRINGS IN PRX FUNCTIONS

Two main ways –

1. Regular-Expression-ID (generated by PRXPARSE function):
2. Perl-Regular-Expression in PRX functions
2. USING TEXT STRINGS IN PRX FUNCTIONS

Two main ways –

1. Regular-Expression-ID (generated by PRXPARSE function):
   - text pattern identifier in numeric number form
   - By passing into PRXPARSE functions.
   - in increment from 1 to n
*PRXPARSE;
/*Code 2a: Understanding how PRXPARSE gives increment of ID in same data step */

Data New;
PRX = prxparse ('/Hospital/');
Put "PRXPARSE: 1st Occurrence of PRXPARSE function in DATA step" PRX=
PRX = prxparse ('/Hospital/');
Put "PRXPARSE: 2nd Occurrence of PRXPARSE function in DATA step" PRX=
PRX = prxparse ('/Hospital/');
Put "PRXPARSE: 3rd Occurrence of PRXPARSE function in DATA step" PRX=
Run;

INSERT Snapshot here
/*Good Programming Practice
Code 2b: PRXPARSE - Common usage in data step */

Data New;
set AE;
*Executing only 1st time when many observations;
retain PRX;
if _N_=1 then PRX = prxparse ('/Nausea/');
*Once string is compiled and returned the value, you can use same for other observations by retaining;
A2=PRXMATCH (PRX, aedecod);
Run;
2. USING TEXT STRINGS IN PRX FUNCTIONS

Two main ways –

• 2. Perl-Regular-Expression in PRX functions
  ◦ Character constant (e.g. ‘/Hospital/’), variable which returns Perl regular expression (PRX)
/*Code 2c: Passing simple string
into PRX functions. */

Data AE1;
Set AE;

*Simply passing regular expressions
into PRX functions;
A2=PRXMATCH ("/Nausea/", aedecod);
Run;
3. MAKING PERL REGULAR EXPRESSIONS

- This is the power of PRX function!!!
- Can be customized and written to search VERY complex text strings (Though we will cover basics, there is a lot to read more!!)
- A Wide variety of meta-characters
I – Slash (/)

- Required!
- Before and after string

<table>
<thead>
<tr>
<th>PRX Expression note</th>
<th>Metacharacter</th>
<th>Syntax (quotation needs to apply when we put in function)</th>
<th>Example of strings</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>With slash</td>
<td>/</td>
<td>/Nausea/</td>
<td>Nausea</td>
<td>Basic expression</td>
</tr>
</tbody>
</table>

/*Code 2c: Passing simple string into PRX functions. */

Data AE1;
Set AE;
*Simply passing regular expressions into PRX functions;
A2=PRXMATCH ("/Nausea/", aedecod);
Run;
# 2 – Pipe (|) - Alternation

- Multiple text string
- Think of it as OR
- If “XXX” OR “YYY”

<table>
<thead>
<tr>
<th>PRX Expression note</th>
<th>Metacharacter</th>
<th>Syntax (quotation needs to apply when we put in function)</th>
<th>Example of strings</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alternation (OR) using Pipe (</td>
<td>)</td>
<td></td>
<td>/Nausea</td>
<td>Vomiting</td>
</tr>
</tbody>
</table>

**2. Simple two or more string constants in one.**

--> Commonly, Programmers use INDEX function multiple times.

**Using ALTERNATION;**

`run;`
**3 – Bracket ([ ]) - Grouping**

- Specific character grouping
- Think of it as OR for specific character

---

<table>
<thead>
<tr>
<th>PRX Expression note</th>
<th>Metacharacter</th>
<th>Syntax (quotation needs to apply when we put in function)</th>
<th>Example of strings</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>With grouping for a specific character</td>
<td>[ ]</td>
<td>&quot;/[Nn]ausea/&quot;</td>
<td>Nausea, nausea</td>
<td>It will match for the character with 1st Character can be capital or small &quot;N&quot;/&quot;n&quot; word nausea</td>
</tr>
</tbody>
</table>

/*Code 3c: PRXMATCH*/

```plaintext
Data AE1;
Set AE;

*3. Simple one string constant as argument. With grouping, programmers can add strings search with more types of words. --> Commonly, Programmers use INDEX function multiple times.

If PRXMATCH ("/[Nn]ausea/", aedecod);
D2=PRXMATCH ("/[Nn]ausea/", aedecod);
run;
```
4 – Character Presence (/w, /s, /d)

- Specific TYPE of character before or after pattern
- Capitalization word makes negation (NOT)

<table>
<thead>
<tr>
<th>PRX Expression note</th>
<th>Metacharacter</th>
<th>Syntax (quotation needs to apply when we put in function)</th>
<th>Example of strings</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>String with ANY ALPHA-NUMERIC, Space, digit character before targeted string</td>
<td>\w, \s, \d</td>
<td>/\w[Nn]ausea/ \s[Nn]ausea /\d[Nn]ausea/</td>
<td>aNausea, Nausea, 1nausea</td>
<td>\w stands for any alpha-numeric character and \w will match a word character (alphanumeric plus &quot;)\s is for the string with a preceding space. \d is for digit</td>
</tr>
</tbody>
</table>

/*Code 3d: PRXMATCH*/
Data AE1;
Set AE;
*******************************************
*4. \w :: String with ANY ALPHA-NUMERIC character around targeted string. E.g. Get strings Nausea or nausea which is preceded by one ALPHA-NUMERIC character.*******************************************;
*If PRXMATCH ("/\w[Nn]ausea/", aedecod) > 0; E2=PRXMATCH ("/\w[Nn]ausea/", aedecod) ; Run;
5 – i - Case Insensitive Search

- To make case insensitive search
- Put at the end

<table>
<thead>
<tr>
<th>PRX Expression note</th>
<th>Metacharacter</th>
<th>Syntax (quotation needs to apply when we put in function)</th>
<th>Example of strings</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Search CASE-INSENSITIVE</td>
<td>/i</td>
<td>/Nausea/i</td>
<td>Nausea, nausea, NAUSEA</td>
<td>Case Insensitive search This will make case insensitive for the targeted string.</td>
</tr>
</tbody>
</table>

```/*Code 3f: PRXMATCH*/

Data AE1;
Set AE;

-----------------------------------------------

*10. \i :: Can I make search CASE-INSENSITIVE. E.g. Get strings Nausea or nausea in any case. The programmer can do grouping. However, the simple way is to put -----------------------------------------------;

K2=PRXMATCH ("s/Nausea/i", aedecod) ;
*If PRXMATCH ("/Nausea/i", aedecod) > 0;
Run;```
6 – Some more…

<table>
<thead>
<tr>
<th>PRX Expression note</th>
<th>Metacharacter</th>
<th>Syntax (quotation needs to apply when we put in function)</th>
<th>Example of strings</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Range of character</td>
<td>[a-z]</td>
<td>/[a-c]ausea/</td>
<td>ausea, bausea, causea</td>
<td>Take character from “a to c” range for 1st character [a-z] will match a character in the range</td>
</tr>
<tr>
<td>Start of the line</td>
<td>^</td>
<td>/^Nausea/</td>
<td>Nausea ….</td>
<td>Only Nausea which is 1st in line ^ will match the beginning of the line</td>
</tr>
<tr>
<td>End of the line</td>
<td>$</td>
<td>/Nausea$/</td>
<td>… Nausea</td>
<td>It will capture only Nausea which is at the end of the line $ will match the end of the line</td>
</tr>
<tr>
<td>Any character</td>
<td>*</td>
<td>/Nausea*/</td>
<td>Nausea /vomiting, Nausea and , Nausea?</td>
<td>Any character after Nausea * can represent no character to any character.</td>
</tr>
</tbody>
</table>
Summary

- PRX Functions are powerful for character manipulations
- With some basic knowledge, programmer can be very effective.
Thanks....

Questions?