

## HOW - 194

---

# An Introduction to Perl Regular Expressions



Ron Cody

## Introduction

---

- Perl regular expressions allow you to describe a text pattern.
- Regular expressions that are available in SAS<sup>®</sup>9 use the same syntax as regular expressions in Perl, a scripting language for UNIX systems.
- You can search for that pattern in a SAS character variable.
- You can extract the text matching the pattern.
- You can replace a located pattern with replacement text.

## Introduction (cont.)

Regular expressions use text and special characters (known as metacharacters) to describe patterns. For example:

<u>Regex</u>	<u>Matches</u>
<code>/cat/</code>	text from a <b>catalog</b>
<code>/\d\d/</code>	numbers <b>23</b> 4 to 567
<code>/\d\d\d-\d\d-\d{4}/</code>	number <b>434-56-9878</b>

## Some Perl Metacharacters



<code>\d</code>	matches any digit
<code>\D</code>	matches a non-digit
<code>\w</code>	matches any word character (any letter, digit, or underscore)
<code>\W</code>	matches any non-word character
<code>\s</code>	matches any white space character
<code>\b</code>	matches a word boundary
<code>[xyz]</code>	matches any one of the characters in brackets
<code>[a-zA-Z]</code>	matches any upper- or lowercase letter
<code>.</code> (period)	matches any single character

## Some Regex Examples

Regular Expression	Matches
<code>/[XYZ]\d\d/</code>	number <b>Y89</b> 123
<code>/\w\w\ \d/</code>	<b>word 9</b> or 10
<code>/\w\w\ \d/</code>	ab123 (no match)
<code>/r[aeiou]n/i</code>	<b>Ronald</b> Cody
<code>/\w\w\w\s\w\w\w/</code>	<b>four letters</b>

### Exercise 1

### Some Perl Metacharacters (cont.)



<code>\(</code>	matches left parenthesis
<code>\.</code>	matches a period
<code>\\</code>	matches a single backslash
<code>x y</code>	matches x or y
<code>^</code>	matches the beginning of a line
<code>\$</code>	matches the end of a line

Examples:

`/Mrs\.|Mr\./` matches Mrs. or Mr.

`^\(d\d\d\)/` matches an area code

`/^x\d/` matches 'x123' but not 'ax123'

---

## Exercise 2

## Repetition Operators

*	matches previous expression zero or more times
+	matches previous expression one or more times
?	matches previous expression zero or one times
{ <i>n</i> }	matches previous expression <i>n</i> times
{ <i>n,m</i> }	matches previous expression at least <i>n</i> times and not more than <i>m</i> times

Examples:

`/cat*/` matches "ca" followed by 0 or more "t's"

`/cat?/` matches "ca" followed by 0 or 1 "t"

`/c(at)?/` matches "c" followed by 0 or 1 occurrences of "at"

`^\d\d+/` matches 2 or more digits

## Understanding the Repetition Operators

Regular Expression: `/ab*c/`

\* = Zero or more times

String	Match
abc	Yes
ac	Yes
abxxc	No
abbc	Yes
abbbc	Yes

## Understanding the Repetition Operators

---

Regular Expression: `/ab+c/`

+ = One or more times

String	Match
abc	Yes
ac	No
abxxc	No
abbc	Yes
abbbc	Yes

---

## Exercise 3

## Character Classes

`[a-zA-Z]` All upper- and lowercase letters

---

`[2-9]` The digits 2-9

`[A-E0-9]` A to E and digits 0-9

`[^xyz]` Not x or y or z

`[^a-e-]` Not a to e or dash

## Demonstrating Repetition Operators

`/x\d{2,4} /`

---

X followed by 2 to 4 digits and a blank

String	Match
--------	-------

X12	Yes
-----	-----

X123	Yes
------	-----

X1234	Yes
-------	-----

X12345	No
--------	----

X123A	No
-------	----

Length of String is 10

# The PRX Functions

---

## The PRXPARSE Function

Function: PRXPARSE

Purpose: Compiles a regular expression

Syntax: PRXPARSE (*expression*)

*expression* is a Perl regular expression.

Examples:

```
Pattern = prxparse("/X\d+/");
```

(matches an uppercase X followed by one or more digits)

Note: To ignore case use **prxparse("/X\d+/i");**

```
RE = prxparse("/\ (8(00|66|77|88)\ ) \d{3}-\d{4}/")
```

(matches a toll-free telephone number starting with 800, 866, 877, or 888)

## The PRXMATCH Function

Function: PRXMATCH

Purpose: Returns the position of a regular expression in a string

Syntax: PRXMATCH (*return-code* or "*regex*", *value*)

*return-code* is the return code from the PRXPARSE function;  
*regex* is a Perl regular expression

*value* is a SAS character value.

If no pattern is found, the function returns a 0.

## The PRXMATCH Function

Examples:

```
Position = PRXMATCH ("/X\d\d/", "ABCX56");
```

Position = 4, start of the pattern

```
Position = PRXMATCH ("/X\d\d/", "Apple");
```

Position = 0, no pattern found

PRXMATCH is like a generalized FIND function

## Task

---

You have a raw data file that contains phone numbers and you want to create a new, temporary SAS data set that contains all toll free numbers (defined as numbers starting with 800, 866, 877, 888).

```
***Solution using only PRXMATCH;
data toll_free;
  input String $15.;
  if prxmatch
    ("/^\(8(00|66|77|88)\)\d{3}-\d{4}/",
    String) then output;
datalines;
(800)123-4555
(908)782-6562
(866)777-8888
(808)131-1311
(877)985-4848
;
```

Toll-free Numbers
String
(800)123-4555
(866)777-8888
(877)985-4848

```
***Solution using PRXPARSE and PRXMATCH;  
data toll_free;  
  input String $15.;  
  
  retain re;  
  if _n_ = 1 then  
    re = prxparse("/^\(8(00|66|77|88)\)\d{3}-\d{4}/");  
  
  if prxmatch(re,String) gt 0 then output;  
datalines;  
(800)123-4555  
(908)782-6562  
(866)777-8888  
(808)131-1311  
(877)985-4848  
;
```

Toll-free Numbers	
string	re
(800)123-4555	1
(866)777-8888	1
(877)985-4848	1

## Task

---

Check that an ID is in the form of an X, Y or Z, followed by one or more digits.

```
data codes;
  Perl = "/[XYZ]\d+/" ;
  input Id $; *Length of Id is 8;
  if prxmatch(Perl,Id) ne 0 then
    Match = 'Yes';
  else Match = 'No';
datalines;
X87
87X77
A567
Z88W
X1234567
;
```

First Try

Perl Regular Expression	Id	Match
/[XYZ]\d+/	X87	Yes
	87X77	Yes
	A567	No
	Z88W	Yes
	X1234567	Yes

```
***Add a beginning of line anchor; )
data codes;
  Perl = "/^[XYZ]\d+/" ;
  input Id $;
  if prxmatch(Perl,Id) ne 0 then
    Match = 'Yes';
  else Match = 'No';
datalines;
X87
87X77
A567
Z88W
X1234567
;
```

Second Try

Perl Regular Expression	Id	Match
/^[XYZ]\d+/	X87	Yes
	87X77	No
	A567	No
	Z88W	Yes
	X1234567	Yes

```

***Add a blank at the end;
data codes;
  Perl = "/^[XYZ]\d+ /";
  input Id $;
  if prxmatch(Perl,Id) ne 0 then
    Match = 'Yes';
  else Match = 'No';
datalines;
X87
87X77
A567
Z88W
X1234567
;

```

Third Try

Perl Regular Expression	Id	Match
/^[XYZ]\d+ /	X87	Yes
	87X77	No
	A567	No
	Z88W	No
	X1234567	No

```

***Add a word boundary at the end;
data codes;
  Perl = "/^[XYZ]\d+\b/";
  input Id $;
  if prxmatch(Perl,Id) ne 0 then
    Match = 'Yes';
  else Match = 'No';
datalines;
X87
87X77
A567
Z88W
X1234567
;

```

Fourth Try

Perl Regular Expression	Id	Match
/^[XYZ]\d+\b/	X87	Yes
	87X77	No
	A567	No
	Z88W	No
	X1234567	Yes

---

## Exercise 4 - 7

---

## Contact Information

Ron Cody

Email: [ron.cody@gmail.com](mailto:ron.cody@gmail.com)