

What are Perl Regular Expressions?

- A tool used for pattern recognition and string manipulation.
- Consists of a sequence of characters that define a pattern, which can be used for tasks like searching, replacing, or splitting text.
- Makes it possible to manipulate data in which the exact values are not predetermined but in which a specific pattern may appear and to search for such patterns.

Why should you use them and not stick to the good old SAS functions?

- For usual string processing tasks but sometimes in a much more compact way than with standard SAS functions:
 - Searching for a specific substring
 - Searching for only alphabetical or numeric characters
 - Replacing a word by another
- For pattern matching and replacement:
 - Searching for a pattern in a variable, like a date in yymmdd10. format
 - Manipulating patterns, like changing the order of first name and last name
 - Replacing a pattern with a fixed value

How do they work?

- Specific syntax and metacharacters to define patterns:
 - **Metacharacters to indicate the type of character expected**
 - **Metacharacters to indicate the position in the string, or to group expressions**
 - **Metacharacters for repetition factors**
- PRX functions in SAS:
 - **PRXPARSE**: compiles a regular expression that can be used later on.
 - **PRXMATCH**: search for and return the position of a regular expression.
 - **PRXCHANGE**: performs a pattern-matching replacement.
 - **PRXPOSN**: Returns a character string that contains the value for a capture buffer.

Examples of Metacharacters

Metacharacter	Description
\d	matches a digit character that is equivalent to [0-9].
\l	specifies that the next character is lowercase.
\s	matches any white space character, including space, tab, form feed, and so on
\w	matches any word character or alphanumeric character, including the underscore.
\	marks the next character as either a special character, a literal, a back reference, or an octal escape
	specifies the 'or' condition when you compare alphanumeric strings
^	matches the position at the beginning of the input string.
\$	matches the position at the end of the input string.
*	matches the preceding subexpression zero or more times
+	matches the preceding subexpression one or more times
?	matches the preceding subexpression zero or one time
{n,m}	m and n are non-negative integers, where n<=m. They match at least n and at most m times
[...]	specifies a character set that matches any one of the enclosed characters
[^...]	specifies a negative character set that matches any character that is not enclosed
[a-z]	specifies a range of characters that matches any character in the range
(?=...)	specifies a zero-width, positive, look-ahead assertion. In the expression 'regex1 (=?regex2)', a match is found if both 'regex1' and 'regex2' match.

Examples of how Perl Regular Expressions can make your life easier

Extracting a date in DDMMYYYY format from a string

TEXTVAR
1 Date is 12NOV1990.
2 No date.

```
data phuse_ex1;
  set phuse_ex1;
  format vardt date9.;
  match=prxmatch("/\d{2}[A-Z]{3}\d{4}/",upcase(textvar));
  if match^=0 then do;
    vardt=input(substr(textvar,match,9),date9.);
  end;
run;
```

TEXTVAR	VARDT
1 Date is 12NOV1990.	12NOV1990
2 No date.	

Extracting numeric results

LBORRES
1 POSITIVE
2 4.32
3 4.3.2
4 <3.2
5 12
6 7u

```
data phuse_ex2;
  set phuse_ex2;
  format lbstresn 8.2;
  if prxmatch("/^\d+\.\?\d*$/",strip(lborres))
  then lbstresn=input(lborres,best.);
run;
```

LBORRES	LBSTRESN
1 POSITIVE	
2 4.32	4.32
3 4.3.2	
4 <3.2	
5 12	12.00
6 7u	

Adding "0" at the beginning of a time variable when time < 12:00

TIME
1 6:50
2 11:25

```
data phuse_ex3;
  set phuse_ex3;
  new_time=prxchange("s/^\d{1}:/0$1/", -1,time);
run;
```

"s/" specifies a substitution regular expression

"0\$1": replaces the previous expression by "0" followed by the first group followed by ":"

TIME	NEW_TIME
1 6:50	06:50
2 11:25	11:25

Searching for multiple substrings

CMDECOD
1 PRESINOL
2 OXYCODONE HYDROCHLORIDE
3 DOPAMIDE
4 ROLATUSS WITH HYDROCODONE
5 LEVOMETHADONE

```
data phuse_ex4;
  set phuse_ex4;
  where prxmatch('/codone|morphone|methadone/i',cmdecod);
run;
```

CMDECOD
1 OXYCODONE HYDROCHLORIDE
2 ROLATUSS WITH HYDROCODONE
3 LEVOMETHADONE