

Perl Regular Expressions and SAS 9



Matthew Rodger
Decision Strategy Analyst
NAB
matthew.d.rodger@nab.com.au

To be covered today



- 1. What are regular expressions?**
2. Perl Implementation
3. SAS Implementation
4. Why use regular expressions?
5. How to use Perl regular expressions in SAS

We have all probably used regular expressions equivalents at some stage.

- ⌘ In UNIX, we have typed `ls -l *.sas` (`dir *.sas` in DOS/Windows). In windows, we have searched for `*.doc`.
- ⌘ The `*` is in a way a regular expression. What we are saying is match any filename with a `.sas`
- ⌘ Likewise, we have probably also typed `ls -l maketable??.sas`, where once again, the `??` means match any two characters. The `?` also works in DOS.
- ⌘ In UNIX/DOS, the `*` and `?` are shell metacharacters
- ⌘ In SQL, we have probably used a query like the following

```
SELECT *
FROM mytable AS a
WHERE a.name LIKE '% RODGER'
```
- ⌘ Once again, the `%` matches zero or many characters, `*` matches all columns

UNIX in particular allowed more flexibility with regular expressions

- ⌘ In UNIX, the grep family of commands allow us to filter using regular expressions on a line. For example to list the sub-directories under the current directory we type the following:

```
ls -la | grep ^d
```
- ⌘ The equivalent DOS command is:

```
dir | findstr/c:DIR
```
- ⌘ The grep command has a lot more functionality mentioned here as well as variant such as egrep and fgrep
- ⌘ The sed command allows us to use regular expressions to change text, for example:

```
ls -l | sed 's/[0-9]{4} //g'
```

To be covered today



1. What are regular expressions?
- 2. Perl Implementation**
3. SAS Implementation
4. Why use regular expressions?
5. How to use Perl regular expressions in SAS

The programming language Perl implements regular expressions.

- ⌘ Frustrated by the restrictions in awk and sed, the great computing guru Larry Wall wrote Perl.
- ⌘ Perl has its own sets of regular expressions, which other languages will adhere to. For example Javascript allows us to use regular expressions for form validation.
- ⌘ In Perl, we could write a sub-routine to match postcodes (seen in SAS later)

```
sub matchcode
{
    my ($pcode)=@_;
    return ($pcode=~ /^[\d]{4}$/) ? "Valid" : "Invalid";
}
```

A non exhaustive list of Perl Regular Expressions.

Metacharacter	Description
*	Matches zero or more cases
+	Matches one or more cases
?	Matches zero or one times
.	Matches one character
^	Matches start of string
\$	Matches end of string
[0-9]	Will match all characters in square brackets
[^0-9]	Don't match characters in square brackets
a b	Matches a or b
{n}	Matches the subpattern n times
{n,}	Matches the subpattern at least n times
{n,m}	Matches the subpattern at least n times, but no more than m times
\d	Matches digits 0-9
\D	Doesn't match digits 0-9
\w	Matches letters and digits as well as _
\W	Opposite to \w
\b	Matches a word boundary
\B	Opposite to \b

To be covered today



1. What are regular expressions?
2. Perl Implementation
- 3. SAS Implementation**
4. Why use regular expressions?
5. How to use Perl regular expressions in SAS

SAS also implements Perl Regular Expressions.



- ⌘ Three functions will be introduced today
 - ☒ PRXPARSE will compile the regular expression
 - ☒ PRXMATCH will match a regular expression and give the position of the match
 - ☒ PRXCHANGE will change text using regular expressions
- ⌘ These are the more commonly used regular expressions and will be extensively covered today. There are other regular expression functions, but these are probably the more common ones used.
- ⌘ SAS has its own regular expression implemented in SAS 6, but Perl regular expressions have been implemented in SAS 9.

SAS also implements Perl Regular Expressions.



⌘ Other useful regular expressions are:

- ☒ PRXSUBSTR this works out the start of the matched expression and the length of it
- ☒ PRXNEXT finds the nth occurrence of the pattern in a string
- ☒ PRXPAREN indicates which of the alternate matches it matched with the largest capture-buffer number
- ☒ PRXPOSN indicates the start and the length of a subexpression defined in a regular expression.

To be covered today



1. What are regular expressions?
2. Perl Implementation
3. SAS Implementation
- 4. Why use regular expressions?**
5. How to use Perl regular expressions in SAS

In a perfect world our data would be clean and easy to query and change



- ⌘ But, we don't live in a perfect world and our data is not clean and we have to live with legacy systems.
- ⌘ SAS has routines to find strings, match strings, change strings. But these are for what I call 'static text'. What about 'dynamic text' like that which could potentially come out of legacy systems?
- ⌘ Regular expressions can help with this. A postcode should be four digits, a telephone number should be 8-10 (I am ignoring international codes) and could include brackets, dashes. Whilst normal SAS procedures can handle this, what about if we are not sure about the number of spaces, the number of dashes. Regular expressions provide a neat way of matching and changing strings.

To be covered today



1. What are regular expressions?
2. Perl Implementation
3. SAS Implementation
4. Why use regular expressions?
- 5. How to use Perl regular expressions
in SAS**

It is time to see these regular expressions in action.



- ⌘ PRXMATCH – let's match an e-mail address
- ⌘ PRXCHANGE

Whilst regular expressions are very useful, the following are tips to remember:



- ⌘ Regular expressions are compiled, that means, you do not want to compile them at every data observations, hence, compile them in the first observation and then RETAIN them.
- ⌘ If an existing SAS string function will perform the task easily, then use it. Use INDEX for matching simple strings and TRANWRD for string replacement.
- ⌘ Regular expressions allow more advanced string matching and should be used for these purposes or for purposes that we would normally have very lengthy and complicated in-built SAS string functions
- ⌘ Regular expressions are hard to interpret, so it is very important to document them, this is not only for others, but for yourself as well!



Thank you for attending