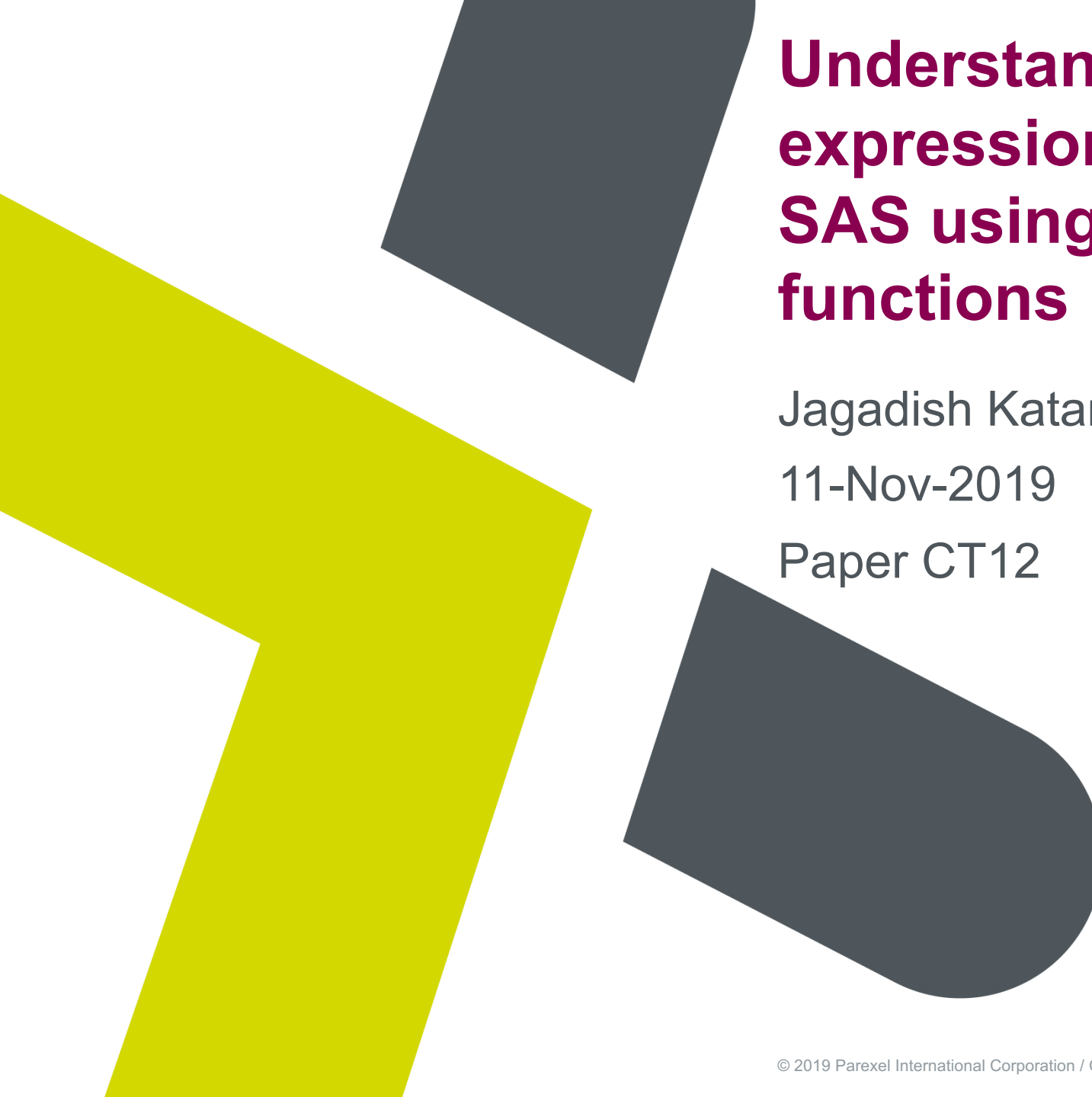


The slide features several large, abstract geometric shapes. A large yellow shape is on the left, and two grey shapes are positioned above and below the main title. The title itself is in a bold, purple font.

Understanding regular expression and its application in SAS using PRX (PERL regEx) functions

Jagadish Katam

11-Nov-2019

Paper CT12

Disclaimer

- The opinions expressed in this presentation and on the following slides are solely those of the presenter and not necessarily those of Parexel. Parexel does not guarantee the accuracy or reliability of the information provided herein.

Agenda

- › Basic Perl Metacharacters
- › SAS Perl Regular Expression Functions
 - › PRXPARSE
 - › PRXMATCH
 - › Comparison of PRXMATCH with INDEXW and FINDW
 - › PRXCHANGE
 - › CALL PRXSUBSTR
 - › PRXNEXT

Basic Perl Metacharacters

- › The following are some of the metacharacters that you can use to match patterns in Perl regular expressions.
- › `\d` - matches a digit character that is equivalent to `[0-9]`.
- › `\w` - matches any “word” character, i.e. alphanumeric or underscore
 - › this is equivalent to `[A-Za-z0-9_]`

Basic Perl Metacharacters

- › A repetition factor matches a preceding subexpression zero or more times as it can when using a specific starting location.
 - › * - matches the preceding subexpression zero or more times
 - › `zo*` matches "z" and "zoo"
 - › + - matches the preceding subexpression one or more times
 - › `"zo+"` matches "zo" and "zoo"
 - › {n,m} - m and n are non-negative integers, where $n \leq m$. They match at least n and at most m times
 - › `"o{1,3}"` matches the first three o's in "foooooood"

Basic Perl Metacharacters

- › By placing part of a regular expression inside round brackets or parentheses, you can group that part of the regular expression together. This allows you to apply a quantifier to the entire group or to restrict alternation to part of the regEx.
- › [a-z] - specifies a range of characters that matches any character in the range
 - › “[a-z]” matches any lowercase alphabetic character in the range “a” through “z”
- › [0-9] - specifies a range of digits that matches any digit in the range
 - › “[0-9]” matches any digit in the range 0 through 9

PRXPARSE and PRXMATCH

- The PRXPARSE function returns a pattern identifier number or pattern that is used by other Perl functions and CALL routines to match patterns. If an error occurs in parsing the regular expression, then SAS returns a missing value. PRXPARSE uses metacharacters in constructing a Perl regular expression.
- The PRXMATCH functions searches for a pattern match and returns the position at which the pattern is found.

PRXPARSE and PRXMATCH

data want;

Set conmed;

```
position=prxmatch('m/MAZEPINE|PHENOBARBITAL|BUTALBITAL|RIFAP/oi',cmdecod);
```

run;

SUBJECT	CMDECOD	POSITION
001	CARBAMAZEPINE	6
001	DOXORUBICIN	0
001	CARBAZINE	0
001	VINCRIStINE	0
002	PILSICAINIDE	0
002	PHENOBARBITAL	1
002	BUTALBITAL	1
002	RIFAP	1

Comparison of PRXMATCH with INDEXW and FINDW

- › The advantages of using the PRXMATCH when compared with regular SAS functions INDEXW and FINDW are:
- › When searching for multiple words like CARBAMAZEPINE, PHENOBARBITAL, BUTALBITAL or RIFAP then we must use INDEXW for each word, it is case sensitive and cannot make case insensitive but searches the complete word.
- › Like INDEXW we can use FINDW for every word, however FINDW has an advantage over INDEXW i.e., it has an option modifier 'i' to ignore the case. But still it needs to be repeated for each word.

Comparison of PRXMATCH with INDEXW and FINDW

› INDEXW

`indexw(cmdecod,'CARBAMAZEPINE')` or `indexw(cmdecod,'PHENOBARBITAL')` or
`indexw(cmdecod,'BUTALBITAL')` or `indexw(cmdecod,'RIFAP')`

› FINDW

`findw(cmdecod,'CARBAMAZEPINE',' ','l')` or `findw(cmdecod,'PHENOBARBITAL',' ','i')` or
`findw(cmdecod,'BUTALBITAL',' ','i')` or `findw(cmdecod,'RIFAP',' ','i')`

› PRXMATCH

`prxmatch('m/CARBAMAZEPINE|PHENOBARBITAL|BUTALBITAL|RIFAP/oi',cmdecod)`

PRXCHANGE

- › Performs a pattern matching replacement.
- › Let's see how we can use the PRXCHANGE to replace the UN and UNK with '--' and '---'.

```
data have;  
input date &$100.;  
updated_date=prxchange('s#(UN\s)(UNK\s)(\d*)#-- --- $3#i',-1,date);  
cards;  
UN UNK 2018  
run;
```

DATE	UPDATED_DATE
UN UNK 2018	-- --- 2018

Metacharacters

- › s – is a tag in prxchange that represents substitution
 - › () - indicates grouping
 - › \s – matches a single space
 - › \d - matches any digit
 - › * - matches preceding subexpression one or more times
-
- › (UN\s) is group 1 which identifies the text UN ending with a space, (UNK\s) is group 2 which identifies the text UNK ending with a space and (\d*) is group 3 which identifies any digit repeated one or more times. '-- --- \$3' \$3 represents (\d*) before which if we keep '-- ---' the text before \$3 group will be replaced with '-- ---'.

CALL PRXSUBSTR

› To extract dose information (DOSE) from the CMDECOD we can use PRXPARSE to get the pattern ID for use in CALL PRXSUBSTR and later use the SUBSTR function. The variables i.e., with starting position (start) and length of the dose information (length) are created by CALL PRXSUBSTR. Here start and length are the variables used in SUBSTR to extract the dose information.

› Sample Data

001 RIFAP 5 mg

001 OXCARBAZEPINE 1000 mg

002 CARBAMAZEPINE

002 PHENYTOIN 62.5 mg

CALL PRXSUBSTR

data want;

set conmed;

```
id=prxparse('/\d[\.\d\s]*\w[mg]*/oi');
```

```
call prxsubstr(id,cmdecod,start,length);
```

```
if start>0 then dose=substr(cmdecod,start,length);
```

run;

SUBJECT	CMDECOD	ID	START	LENGTH	DOSE
001	RIFAP 5 mg	1	7	4	5 mg
001	OXCARBAZEPINE 1000 mg	1	15	7	1000 mg
002	CARBAMAZEPINE	1	0	0	
002	PHENYTOIN 62.5 mg	1	11	7	62.5 mg

Metacharacters

- › `\d` – matches any digit
- › `[ ]` – matches any character or metacharacter within `[ ]` example `[\.\d\s]` matches a dot or digit or single space
- › `*` - matches preceding subexpression one or more times
- › `\w` – matches a single letter, when used like `\w[mg]` – matches a letter with 'm' or 'g'

CALL PRXNEXT

- › Returns the position and length of a substring that matches a pattern and iterates over multiple matches within one string.

- › **Sample data**

```
data conmed;  
input subject$ resupply&$300.;  
cards;  
001 10 vials for this patient send 13/12/09  
002 04/2/12 - 240 mg 1/2/12 - 240 mg 15/2/12 - 240 mg 22/2/12 - 240 mg  
003 1st Re-supply - 1/02/2013(medicine) 2nd Re-supply - 1/07/2013(medicine)  
004 1st re-supply - 03-FEB-2015 - medicine - 11 vials 2nd re-supply 11-Feb-2015 medicine - 1 vial  
005 1st re-supply - 03/JAN/2016 - medicine - 11 vials 2nd re-supply 11/JAN/2016 medicine - 1 vial  
006 1st re-supply - 03/MAR/16 - medicine - 11 vials 2nd re-supply 11/MAR/16 medicine - 1 vial  
007 resupplied as medicine on 9 Jan 10- same dose  
run;
```

CALL PRXNEXT

```
data want;  
set conmed;  
start=1;  
stop=length(resupply);  
pattern_id=prxparse('/\d{1,2}[\V-]\d{1,2}[\V-]\d{2,4}|\d{1,2}[(\s)?(\s)?\,V]\w{3,}[(\s)?(\s)?V]\d{2,4}\b/');  
call prxnext(pattern_id,start,stop,resupply,position,length);  
do while(position>0);  
  resupply_date=substr(resupply,position,length);  
  call prxnext(pattern_id,start,stop,resupply,position,length);  
output;  
end;  
run;
```

Metacharacters

SUBJECT	RESUPPLY	START	STOP	PATTERN_ID	POSITION	LENGTH	RESUPPLY_DATE
001	10 vials for this patient send 13/12/09	40	39	1	0	0	13/12/09
002	04/2/12 - 240 mg 1/2/12 - 240 mg 15/2/12 - 240 mg 22/2/12 - 240 mg	24	66	1	18	6	04/2/12
002	04/2/12 - 240 mg 1/2/12 - 240 mg 15/2/12 - 240 mg 22/2/12 - 240 mg	41	66	1	34	7	1/2/12
002	04/2/12 - 240 mg 1/2/12 - 240 mg 15/2/12 - 240 mg 22/2/12 - 240 mg	58	66	1	51	7	15/2/12
002	04/2/12 - 240 mg 1/2/12 - 240 mg 15/2/12 - 240 mg 22/2/12 - 240 mg	58	66	1	0	0	22/2/12
003	1st Re-supply - 1/02/2013(medicine) 2nd Re-supply - 1/07/2013(medicine)	62	71	1	53	9	1/02/2013
003	1st Re-supply - 1/02/2013(medicine) 2nd Re-supply - 1/07/2013(medicine)	62	71	1	0	0	1/07/2013
004	1st re-supply - 03-FEB-2015 - medicine - 11 vials 2nd re-supply 11-Feb-2015 medicine - 1 vial	76	93	1	65	11	03-FEB-2015
004	1st re-supply - 03-FEB-2015 - medicine - 11 vials 2nd re-supply 11-Feb-2015 medicine - 1 vial	76	93	1	0	0	11-Feb-2015
005	1st re-supply - 03/JAN/2016 - medicine - 11 vials 2nd re-supply 11/JAN/2016 medicine - 1 vial	76	93	1	65	11	03/JAN/2016
005	1st re-supply - 03/JAN/2016 - medicine - 11 vials 2nd re-supply 11/JAN/2016 medicine - 1 vial	76	93	1	0	0	11/JAN/2016
006	1st re-supply - 03/MAR/16 - medicine - 11 vials 2nd re-supply 11/MAR/16 medicine - 1 vial	72	89	1	63	9	03/MAR/16
006	1st re-supply - 03/MAR/16 - medicine - 11 vials 2nd re-supply 11/MAR/16 medicine - 1 vial	72	89	1	0	0	11/MAR/16
007	resupplied as medicine on 9 Jan 10- same dose	35	45	1	0	0	9 Jan 10

- › `\d{1,2}[\V-]\d{1,2}[\V-]\d{2,4}` - 4/26/2013 or 28/10/09
- › `\d{1,2}[(\s)?(\s)?\s,V]\w{3,}[(\s)?(\s)?V]\d{2,4}` - 14-Mar-2016 or 9 Dec 10 or 11/Feb/2016

Conclusion

- The examples provided here would help the programmers understand how they can be used in the day to day programming as an alternate to regular SAS functions like INDEXW, FINDW, SUBSTR etc.,
- The metacharacters listed in this paper are only those which are used in the examples, there are a lot of other metacharacters which are available online.
- We covered PRXMATCH, PRXCHANGE and CALL SUBSTR with some examples. We also saw the advantage of call routine CALL PRXNEXT in its robustness to fetch different patterns of dates from the complex text data.

Paper

- There is more in the paper than can be presented in 8 minutes, so please refer to my paper CT12 for detailed information and please feel free to contact for any questions.

References

- › Google search of SAS documentation website for metacharacters and PRX functions.
- › Referred regular expressions website for different metacharacters:
<https://www.regular-expressions.info/>
- › Windham, K. Matthew. 2014. Introduction to Regular Expressions in SAS®. Cary, NC: SAS Institute Inc.



Thank you
Any questions?

parexel[®]