PharmaSUG 2015 - Paper BB14

Perl Regular Expression in SAS® Applications

Yang Wang, Seattle Genetics, Inc., Bothell, WA
Abdul Ghouse, Seattle Genetics, Inc., Bothell, WA

ABSTRACT

SAS® provides text functions to perform search, replace and other manipulations on character strings. SAS® also provides a set of functions that works with Perl regular expression (PRX functions) to perform string tasks with more flexibility and power. In this paper, we use applications or macro examples to demonstrate advantages of the PRX functions:

An automated emailing utility: A SAS® application that automates a process that selectively sends out report files to different email addresses based on a specific schedule. User-defined strings of Perl regular expression used to select the target report files are maintained in a lookup file.

A smart Excel® reader: Excel® allows mixed data type in one column but SAS® dataset column only allows a single data type. In order to read in all the cells in their intended format, this SAS® macro identifies the cell format by PRX functions and casts the right data type for the column.

An efficient SAS® macro input parameters checker: We provide examples to show PRX functions have great advantages over SAS® text functions.

In conclusion, because Perl regular expression patterns can be written in the form of static string, it allows users to pass static patterns that provide logic to a SAS® program instead of nesting one or more string functions.

INTRODUCTION

Perl stands for Practical Extraction and Report Language. It is a very powerful tool to extract information from text files and generate reports. It is widely used as a script language for Unix administration and Web programming. The main component of its power is pattern matching using a technique called regular expression. Regular Expression is a technical term for a simple way of doing pattern matching. You can use regular expression to look at data and process the data in a very efficient way.

Perl regular expression was introduced to SAS from version 9. There are multiple SAS functions designed to utilize Perl regular expression, two of them are used the most often: PRXPARSE and PRXMATCH. These two functions are used in the three application examples in this paper.

An Example to demonstrate how the two functions are used together to find matching patterns

```
Data FindFriend;
    Set checkfile;
    If _n_ = 1 then do;
          Retain regex;
          Regex = prxparse ('/Friend/');
          If prxmatch (regex, relationship);
Run;

/*Or combine the two function into one*/
Data FindFriend;
    Set checkfile;
    If _n_ = 1 then do;
         If prxmatch ('/Friend/', relationship);
Run;
```

Variable 'regex' contains the matching value for Regular expression which is the string 'Friend'. It is put inside slashes in function calls. Function PRXMATCH will return the position in the string where the match starts. In the above example, if dataset variable “relationship’ contains the string ‘Friend’, the record will be stored in the dataset FindFriend.

In the example above, regular expression is string ‘Friend’. More frequently, matching pattern is used in programming because it is flexible and effective. For example, pattern /^\d+/ will match any string that starts with one or more digits; /b*r+z/ will match any of the following strings:

br

brz

bbrrz

bbbbbbbrrrrrrz

Because SAS provides powerful string functions that can find and replace strings in text files, Perl regular expression is not often used by many programmers. We present this paper to demonstrate that in some applications, it is more efficient to use Perl regular expression. SAS programmers are encouraged to utilize Perl regular expression functionalities.

EXAMPLE 1: AUTOMATED EMAILING UTILITY

The Report Distribution Utility automates the distribution of periodic safety reports to internal and external clients. The utility consists of the following components:

- A excel file that defines the details (content, schedule, recipients, etc.) for each period deliverable.
- An HTML application that allows users to select reports for delivery by providing a user interface to the utility.
- A SAS program that transfers the files from the SAS fileshare to the shared server and email distribution server.
- Software that reads email requirements from SAS and generates emails to internal and external recipients via corporate exchange server.
- A command-line sFTP tool that transfers SAS datasets to external recipients.

The application reads in the database file that identifies the available reports and their standard distribution timelines. The user can send the reports due for that day, or select a custom subset of reports. When the user clicks “Send” in the user interface, the SAS program moves the report objects to the shared server and email server. The SAS program calls the email generation software via PROC HTTP, passing the definition of each email’s content. The application creates the email with attachments for external recipients, and attachments and/or hyperlinks to the shared server for internal recipients. A log detailing the distribution is generated, and the status (success/failure) of each report distribution is relayed back to HTML application user interface. Figure 1 is the illustration of the process.

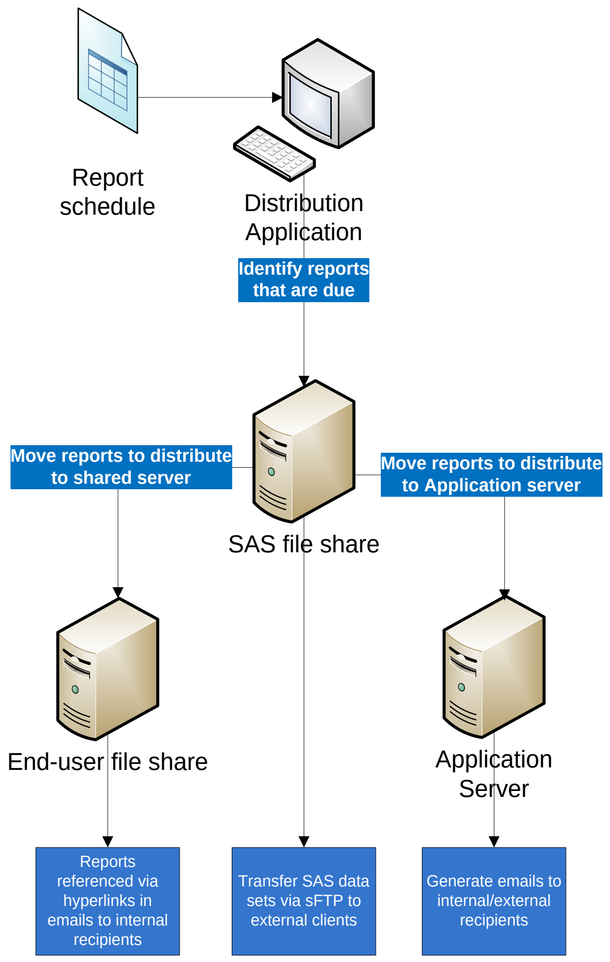
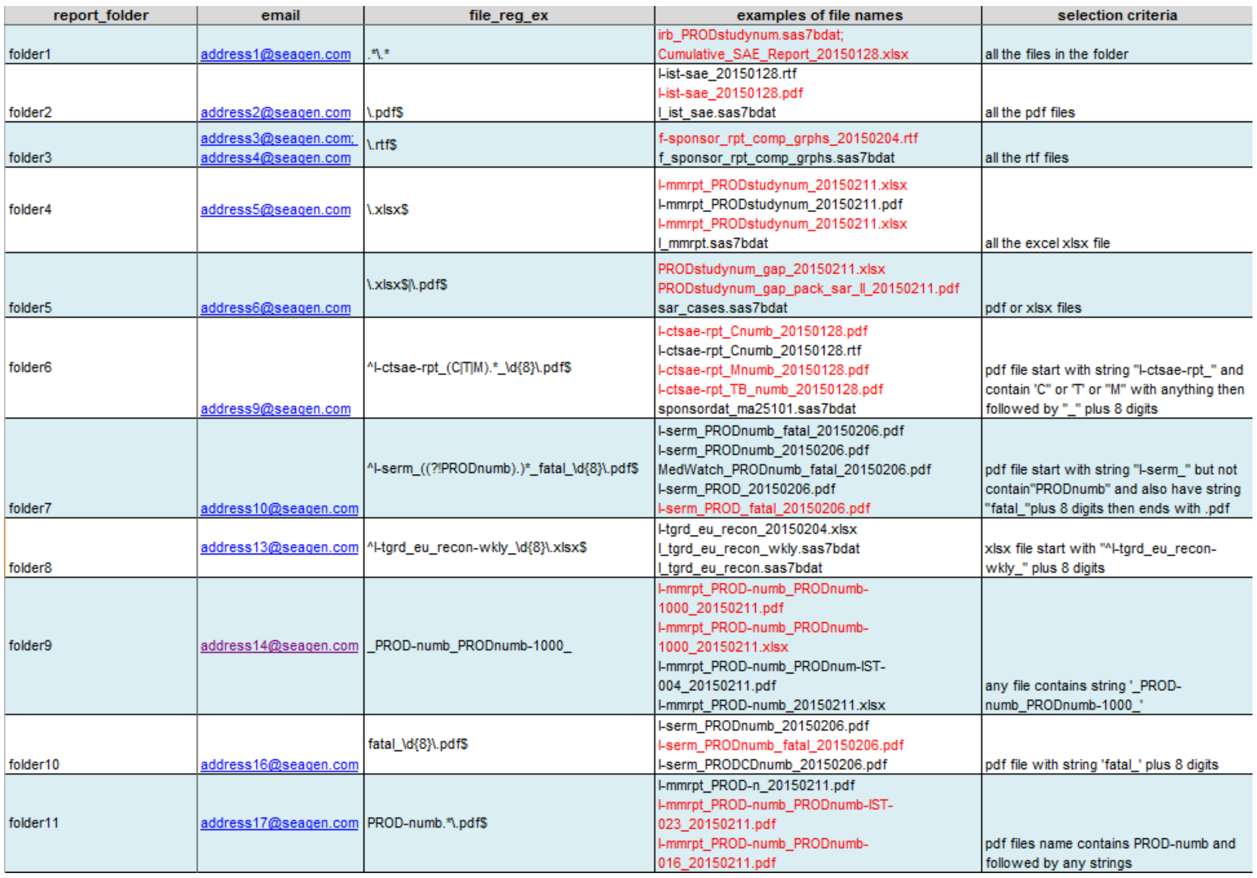

Report schedule
Distribution Application
Identify reports that are due
Move reports to distribute to shared server
SAS file share
Move reports to distribute to Application server
End-user file share
Application Server
Reports referenced via hyperlinks in emails to internal recipients
Transfer SAS data sets via sFTP to external clients
Generate emails to internal/external recipients

Figure 1.Automated report distribution process

The Report Schedule in Figure 1 is an excel file. Column "Report Folder" points to the folder contains the report files, Column file_reg_ex contains Perl Regular Expressions that is used to select files in a folder to be sent out. Only the filenames that match the Perl Regular Expression pattern marked in red will be selected and sent out to the email addresses in this process.



report_folder	email	file_reg_ex	examples of file names	selection criteria
folder1	address1@seagen.com	.*\.*	irb_PRODstudynum.sas7bdat; Cumulative_SAE_Report_20150128.xlsx	all the files in the folder
folder2	address2@seagen.com	\.pdf$	l-ist-sae_20150128.rtf l-ist-sae_20150128.pdf l_ist_sae.sas7bdat	all the pdf files
folder3	address3@seagen.com; address4@seagen.com	\.rtf$	f-sponsor_rpt_comp_grphs_20150204.rtf f_sponsor_rpt_comp_grphs.sas7bdat	all the rtf files
folder4	address5@seagen.com	\.xlsx$	l-mmrpt_PRODstudynum_20150211.xlsx l-mmrpt_PRODstudynum_20150211.pdf l-mmrpt_PRODstudynum_20150211.xlsx l_mmrpt.sas7bdat	all the excel xlsx file
folder5	address6@seagen.com	\.xlsx$\|\.pdf$	PRODstudynum_gap_20150211.xlsx PRODstudynum_gap_pack_sar_ll_20150211.pdf sar_cases.sas7bdat	pdf or xlsx files
folder6	address9@seagen.com	^l-ctsae-rpt_(C|T|M).*_\d{8}\.pdf$	l-ctsae-rpt_Cnumb_20150128.pdf l-ctsae-rpt_Cnumb_20150128.rtf l-ctsae-rpt_Mnumb_20150128.pdf l-ctsae-rpt_TB_numb_20150128.pdf sponsordat_ma25101.sas7bdat	pdf file start with string "l-ctsae-rpt_" and contain 'C" or 'T' or "M" with anything then followed by "_" plus 8 digits
folder7	address10@seagen.com	^l-serm_((?!PRODnumb).)*_fatal_\d{8}\.pdf$	l-serm_PRODnumb_fatal_20150206.pdf l-serm_PRODnumb_20150206.pdf MedWatch_PRODnumb_fatal_20150206.pdf l-serm_PROD_20150206.pdf l-serm_PROD_fatal_20150206.pdf	pdf file start with string "l-serm_" but not contain"PRODnumb" and also have string "fatal_"plus 8 digits then ends with .pdf
folder8	address13@seagen.com	^l-tgrd_eu_recon-wkly_\d{8}\.xlsx$	l-tgrd_eu_recon_20150204.xlsx l_tgrd_eu_recon_wkly.sas7bdat l_tgrd_eu_recon.sas7bdat	xlsx file start with "^l-tgrd_eu_recon-wkly_" plus 8 digits
folder9	address14@seagen.com	_PROD-numb_PRODnumb-1000_	l-mmrpt_PROD-numb_PRODnumb-1000_20150211.pdf l-mmrpt_PROD-numb_PRODnumb-1000_20150211.xlsx l-mmrpt_PROD-numb_PRODnum-IST-004_20150211.pdf l-mmrpt_PROD-numb_20150211.xlsx	any file contains string '_PROD-numb_PRODnumb-1000_'
folder10	address16@seagen.com	fatal_\d{8}\.pdf$	l-serm_PRODnumb_20150206.pdf l-serm_PRODnumb_fatal_20150206.pdf l-serm_PRODCDnumb_20150206.pdf	pdf file with string 'fatal_' plus 8 digits
folder11	address17@seagen.com	PROD-numb.*\.pdf$	l-mmrpt_PROD-n_20150211.pdf l-mmrpt_PROD-numb_PRODnumb-IST-023_20150211.pdf l-mmrpt_PROD-numb_PRODnumb-016_20150211.pdf	pdf files name contains PROD-numb and followed by any strings

Figure 2. File names, email address and Perl regular expression is stored in the excel database.

The code that uses Perl regular expression to select files is shown below.

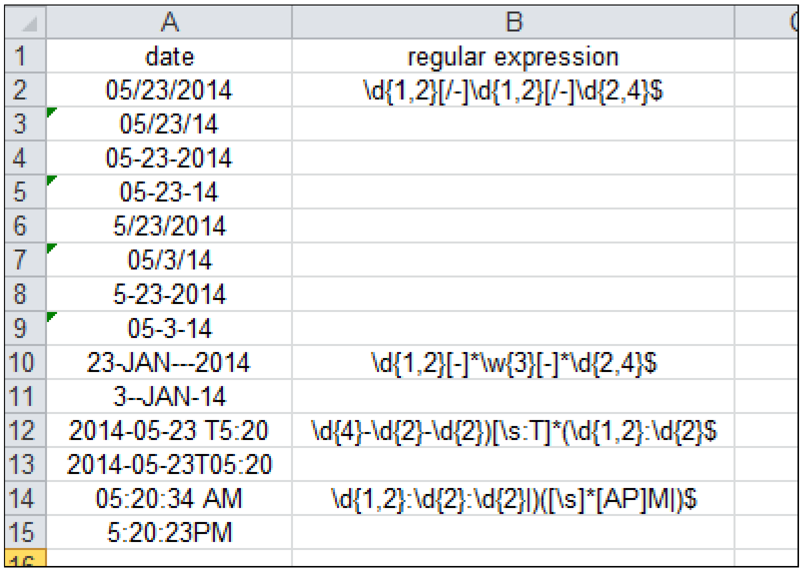
```
rgx=prxparse('/&file_reg_ex/i');
if prxmatch( rgx, strip(filename)) then do;
    ****send out email;
end;
```

The output files generated in the folder are data driven, that is after a new study is added to the database additional outputs will be generated. All the output names are based on the study name stored in the database.

We decided to use Perl Regular Expression to select the files by pattern matching, because this strategy has more advantages over SAS string functions. First, we prefer to set up a look up table to manage the folders and file selection. By using a look up table we can avoid going to the program to update code every time we add new folders or if new outputs are created within a folder. Second, if we aim to put the selection criteria into one column, and perform the same function call on this column to do the file selection in all the folders. If we use SAS string functions, most of the folders will need unique string function calls to perform the file selection, every time a folder or additional outputs are generated, we will need to update SAS code. Perl regular expression is also used to validate email address.

EXAMPLE 2: A SMART EXCEL READER

Excel® allows mixed data type in one column but SAS® dataset column only allows a single data type. In order to read in all the cells in their intended format, this SAS® macro identifies the cell format by PRX functions and casts the right data type for the column.



	A	B
1	date	regular expression
2	05/23/2014	\d{1,2}[/-]\d{1,2}[/-]\d{2,4}$
3	05/23/14	
4	05-23-2014	
5	05-23-14	
6	5/23/2014	
7	05/3/14	
8	5-23-2014	
9	05-3-14	
10	23-JAN---2014	\d{1,2}[-]*\w{3}[-]*\d{2,4}$
11	3--JAN-14	
12	2014-05-23 T5:20	\d{4}-\d{2}-\d{2})[\s:T]*(\d{1,2}:\d{2}$
13	2014-05-23T05:20	
14	05:20:34 AM	\d{1,2}:\d{2}:\d{2}|)([\s]*[AP]M|)$
15	5:20:23PM	

Figure 3. Possible datetime entry in excel files. All the above formats will be reformatted into SAS datetime datatype.

Code used to match the date entry pattern:

```
Data dateset1;
    Set dataset0;
    Rgx=prxparse("/
       (\d{1,2}[/-]\d{1,2}[/-]\d{2,4}|
        \d{1,2}[-]*\w{3}[-]*\d{2,4}|
        \d{4}\d{2}\d{2})[\s:T]*(\d{1,2}:\d{2}|
        \d{1,2}:\d{2}:\d{2}|)([\s]*[AP]M|)$
        /");

    If prxmatch (Rgx, date) then do;
    /* it is a date format, write them into sas date */
    End;
Run;
```

Figure 3 is an example when dates are entered in different text format by user, we will use a program to rewrite valid dates into SAS date format. One line of code with Perl Regular expression can be used to recognize and confirm all the entries are valid dates and then convert them into valid SAS datetime using a customized function in .net. If we use SAS string functions to validate the date and time entries, the code will be much longer.

EXAMPLE 3: AN EFFICIENT SAS® MACRO INPUT PARAMETERS CHECKER

```
%macro  mcrexcel_to_sas( xls=
                     ,headrow=1
                     ,outlib=work
                     ,sheets=1
                     ,outdsn=NONE
                     ,typecast=1
                     ,maxcol=0
                     ,columns=ALL
                     ,varnames=_DFLT_
                     ,quickRead=0
                       );

*Typecast must be on of 1 or 0
if (prxmatch(/^[01]$/i, typecast))<1 then do;
put (ERR)OR: typecast must be either 1  or 0;
end;

* Maxcol must be 0 or an Excel column name
if prxmatch(/^([A-Z]{1,2}|0)$/i, maxcol))<1 then do;
put ERR)OR: MAXCOL must be the name of an Excel column or 0, to indicate
all columns;
end;
```

Example 3 includes a macro call and two blocks of code using Perl Regular Expression to check user input for macro parameters. The first block checks variable “typecast” to be 1 or 0, this can be easily done without Perl Regular expression. The second block checks if the user input is digit 0 or any valid Excel column name, which include A, B, C…Z, AA, AB, AC…ZA, ZB....ZZ. A simple Perl Regular Expression can accomplish this task in a very elegant way.

CONCLUSION AND DISCUSSION

There are more SAS PRX functions available in SAS 9 DATA step that utilize Perl Regular Expression to do text match and substitute: PRXCHANGE, PRXSUBSTR, PRXPOSN, PRXNEXT etc. We only used PRXPARSE and PRXMATCH in the examples. We encourage users to explore them.

SAS has many powerful functions to perform search and string manipulations. However, Perl Regular expression is known for its simplicity and power. The code with Perl Regular Expression usually is easier to read and maintain once you are familiar with it. Pattern matching is also commonly used in other languages such as .net. SAS programmers should take full advantage of this tool.

ACKNOWLEDGMENTS

We thank Shawn Hopkins for sharing the code and valuable suggestions

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Name: Yang Wang
Enterprise: Seattle Genetics, Inc.
Address: 21823 30th Drive Southeast
City, State ZIP: Bothell, WA 98021
E-mail: yawang@seagen.com

Name: Abdul Ghouse
Enterprise: Seattle Genetics, Inc.
Address: 21823 30th Drive Southeast
City, State ZIP: Bothell, WA 98021
E-mail: aghouse@seagen.com