PharmaSUG 2017 - Paper QT04

Is There a Date in That Comment? Use SAS® to Find Out.

Keith Hibbetts, Eli Lilly and Company, Indianapolis, IN

ABSTRACT

When anonymizing data from clinical trials, it's critical to identify any dates contained in the data. This can be a challenge when dealing with data that came from free form text entry, such as comments, where a date could be one small piece of a larger text string. This paper will show how utilizing Perl Regular Expressions in SAS® code can identify records that likely contain a date.

INTRODUCTION

In today's environment, many pharmaceutical companies choose to make data from clinical trials available to external research organizations. In doing so, a company must carefully de-identify the data in order to ensure patient privacy. Dates that appear in the data are an important consideration in de-identification. Typically, date or datetime variables will have a consistent, randomly generated, offset applied throughout the study. One must also consider the possibility that variables created from free text entry, such as comments, could have references to dates as well. The date reference may appear in the middle of the text. Manual assessment of every record of a free text variable would be very time consuming. This paper will demonstrate how Perl Regular Expressions in SAS® code can be used to identify records that likely contain a date.

SAMPLE CODE

Consider the following example: you need to check the CO dataset to see if COVAL (a free text comment) contains any references to dates on the trial. The following lines show samples of COVAL values in the dataset:

> Subject was late to visit due to car trouble
>
> Subject may have to switch sites due to job change
>
> Dose was missed in March due to AE. Makeup dose will be dispensed
>
> Makeup dose was dispensed on 25Apr2014
>
> On Oct-05-2014 vitals were re-taken due to equipment malfunction.
>
> Subject withdrew on 04.12.2014 due to a family emergency.

In the above example, the first two comments have no reference to dates. These would require no de-identification. The third example explicitly references the month of March. This may or may not require de-identification, depending on your organization's guidelines. The fourth through sixth examples explicitly reference full dates, and must be de-identified. These three records also demonstrate that there are multiple ways that a date can be represented in text. Manually looking at 6 records to determine if dates are included is easy. But if the dataset were large (100,000 records, for example) the manual effort would be unwieldy. Perl Regular Expressions can help by sub-setting the dataset to only records where COVAL likely contains a date. Below is a sample of code to perform this logic.

```
DATA co_test;
Set co;
  date_search =
prxparse("m*january|february|march|april|june|july|august|september|october
|november|december|jan\.|feb\.|mar\.|apr\.|may\.|jun\.|jul\.|aug\.|sep\.|oc
t\.|nov\.|dec\.|, 19|, 20|\d-
|\djan|\dfeb|\dmar|\dapr|\dmay|\djun|\djul|\daug|\dsep|\doct|\dnov|\ddec|ja
n,|feb,|mar,|apr,|may,|jun,|jul,|aug,|sep,|oct,|nov,|dec,|jan \d|feb \d|mar
\d|apr \d|may \d|jun \d|jul \d|aug \d|sep \d|oct \d|nov \d|dec
```

```
\d|jan\d|feb\d|mar\d|apr\d|may\d|jun\d|jul\d|aug\d|sep\d|oct\d|nov\d|dec\d|
\d/|\d\d \d\d \d\d|\d:\d|\d\.\d\d\.\d|\d\.\d\.\d|jan-|feb-|mar-|apr-|may-
|jun-|jul-|aug-|sep-|oct-|nov-|dec-*oi");
  if prxmatch(date_search,coval)>0 then output;
run;
```

One note about the code above: it will generate a warning message in the log file. The text of the warning message is listed below. This message can be ignored. The message is generated because the length of the Perl Regular Expression is greater than 262 characters:

```
WARNING: The quoted string currently being processed has become more than
262 characters long.  You may have unbalanced quotation marks.
```

FUNCTIONS USED IN THE CODE

In the code above, there are two functions used.

1. PRXPARSE – This function compiles the PERL Regular Expression and returns a pattern identifier number that can be used by other PERL functions (such as PRXMATCH). In this example, PRXPARSE creates the regular expression id DATE_SEARCH. This describes the pattern(s) that will be searched for by PRXMATCH.
2. PRXMATCH – This function has two arguments: the regular expression id (in this example, DATE_SEARCH) and the source (in this example, COVAL). PRXMATCH will search the source (COVAL) and return the position of the first text in source matching the pattern(s) described in the regular expression id. If no matching patterns are found, PRXMATCH will return a value of 0.

   if prxmatch(date_search,coval)>0 then output;

   The line of code directly above then outputs only records where the value of COVAL contains a pattern described in DATE_SEARCH.

BREAKING DOWN THE PERL REGULAR EXPRESSION

In this section of the paper, we will break down the components of DATE_SEARCH to illustrate how it identifies likely references to dates in COVAL. Let's look at the beginning and end of the expression first, then explore the bulk of the expression.

```
m*
```

This section of the expression indicates that we are building a matching string. The asterisk (*) is being used to define the beginning and end of the matching string. Any non-numeric character can be used for this task. I chose the asterisk because it is not used in anywhere else in the expression.

```
*oi
```

This is the end of the expression. The asterisk (same non-numeric character used at the beginning of the expression) indicates the end of the expression. The o character indicates that the regular expression should only be compiled once (thus saving processing time). The i character indicates that case should be ignored when matching.

```
january|february|march|april|june|july|august|september|october|november|december
```

This is the first section of the expression that is defining patterns to look for in text. The vertical bar character (|) is a delimiter between values. This section lists the full name of months of the year. You might notice that May has been omitted. This was done purposely, as unlike other months May shares a spelling with a commonly-used English word. Including May would significantly increase the number of "false positives"

```
jan\.|feb\.|mar\.|apr\.|may\.|jun\.|jul\.|aug\.|sep\.|oct\.|nov\.|dec\.
```

The combination "\." is used to indicate a period (.). This section of the expression searches for the abbreviation for each month followed by a period, such as "Feb.".

```
, 19|, 20
```

This section of the expression searches for instances where a comma is followed by a space and then either "19" or "20". This search will identify situations that may contain the reference to a year. Examples would be "February 25, 1996" or "September 14, 2007".

```
\d-
```

The combination "\d" indicates a number. This section of the expression will locate instances where a number is followed by a dash. This can help identify situations where a date is expressed in formats such as DDMMYYD (17-03-14) or DDMMYYD10 (17-03-2014).

```
\djan|\dfeb|\dmar|\dapr|\dmay|\djun|\djul|\daug|\dsep|\doct|\dnov|\ddec
```

This section of the expression will locate instances where a number is followed by a month abbreviation. This can help identify situations where a date is expressed in formats such as DATE (01APR14) or DATE9 (19Mar1981).

```
jan,|feb,|mar,|apr,|may,|jun,|jul,|aug,|sep,|oct,|nov,|dec,
```

This section of the expression will locate instances where a month abbreviation is followed by a comma. This can help identify situations where a date is written as, for instance, "Apr, 2014".

```
jan \d|feb \d|mar \d|apr \d|may \d|jun \d|jul \d|aug \d|sep \d|oct \d|nov \d|dec \d
```

This section of the expression will locate instances where a month abbreviation is followed by a space and then a number. This can help identify situations where a date is written as, for instance, "Jan 5" or "Jan 2005".

```
jan\d|feb\d|mar\d|apr\d|may\d|jun\d|jul\d|aug\d|sep\d|oct\d|nov\d|dec\d
```

This section of the expression is very similar to the section directly above, but will locate instances where the month abbreviation is followed directly by a number, such as "Jan5".

```
jan-|feb-|mar-|apr-|may-|jun-|jul-|aug-|sep-|oct-|nov-|dec-
```

This section of the expression will locate instances where a month abbreviation is followed by a dash, such as "Jan-".

```
\d/
```

This section of the expression will locate instances where a number is followed by a forward slash. This can help locate instances where dates are expressed in formats such as DDMMYY (17/03/99) or MMDDYY (03/17/99).

```
\d\d \d\d \d\d
```

This section of the expression will locate instances where two numbers are followed by a space, two additional numbers, another space, and then two additional numbers. This can help identify situations where a date is written as, for instance, "03 17 99".

```
\d:\d
```

This section of the expression will locate instances where a number is followed by a colon and then another number. This can help identify situations where a specific time is included in the text, which might be accompanied by a date.

```
\d\.\d\d\.\d|\d\.\d\.\d
```

This section of the expression will locate instances where a number is followed by a period, either one or two number digits, another period, and then another number. Examples of this format of date would include "2.2.2014" or "3.31.1999".

CONCLUSION

Identifying whether a text string contains a reference to a date is a time consuming activity, and the code discussed here can help reduce the time needed to examine the data to look for potential dates. As well, I hope this paper inspires readers to learn more about using Perl Regular Expressions in SAS® code.

REFERENCES

Eberhardt, Peter and Liu, Wei (Wesley). 2013. "The Baker Street Irregulars Investigate: Perl Regular Expressions and CDISC." Proceedings of the PharmaSUG 2013 Conference. Available at http://www.lexjansen.com/pharmasug/2013/BB/PharmaSUG-2013-BB02.pdf

CONTACT INFORMATION

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

Keith Hibbetts
Eli Lilly and Company
Keith.Hibbetts@lilly.com