SESUG 2020 Paper 104

Check the aCRF through Programming

Hengwei Liu, Daiichi Sankyo, Inc.

ABSTRACT

The annotated case report form (aCRF) is a very important document for the creation of SDTM datasets. Study programmer needs to make sure that the aCRF is accurate, up-to-date and consistent with the SDTM data specifications and SDTM datasets.

Presented in this paper is a programming approach to compare different versions of aCRF for the same study, and to compare the aCRF with the SDTM data specifications and SDTM datasets.

INTRODUCTION

Many pharmaceutical companies have the SDTM datasets created by some contract research organization (CRO). The programmers in the CRO create the aCRF and the SDTM data specifications. The sponsor programmer reviews these documents and provides comments. The CRO programmers make update accordingly.

The sponsor programmer needs to check the aCRF to ensure the updates being made are the same as requested. The aCRF is part of the FDA submission package so mistakes in the document are not acceptable. But it is tedious and time-consuming to check the aCRF page by page. Programming check could significantly improve efficiency. To this end, both SAS® and Python programs can be used.

The readers of this article are expected to understand regular expressions. There is an excellent introduction to this topic (Cody 2004).

The work can be done in three steps:

- Create a dataset containing page number and form name from aCRF.
- Create a dataset containing page number and annotations from aCRF.
- Merge the two datasets together by page number.

To make this process a success, the programmer doing the aCRF annotations must follow certain rules. When an SDTM variable is referred to, the format DOMAIN.VARIABLE must be used. When the value of QNAM in the supplemental domain is displayed, the format SUPPXX.QNAM= must be used. These rules are required so that regular expressions can be used to find the variable names and QNAM values.

GET INTO THE DETAILS

A three-page aCRF in the appendix is used to illustrate the process.

STEP 1

Select all from the aCRF.pdf, copy the content into notepad and save it as acrf.txt.

Display 1 shows a sample of the acrf.txt.

```
Case Report Forms
Study XXXXX-XXXXX-001
1 of 3
Form: Date of Visit
Date of Visit DD MMM YYYY
_______________
2 of 3
Form: Informed Consent
Date Subject or Legally authorized person signed the informed consent DD MMM
YYYY
_____________
Protocol Version Consented under 1.0
2.0
3.0
4.0
3 of 3
Form: Demographics
Date of Birth DD MMM YYYY
___________ Sex Male
Female
Race White
Black or African American
Asian
American Indian or Alaska Native Native Hawaiian or Other Pacific islander
Other
Race other, specify _____________________________________
```

Display 1. Sample of acrf.txt

The SAS function PRXMATCH is used to extract the page number and form name from acrf.txt. The program is get_form.sas.

```
** get_form.sas ** ;

data acrf;
infile "/xxxx/xxxx/acrf.txt" print lrecl=200 pad missover;
input text $char200.;
if prxmatch("/^Form:/", text)>0 or prxmatch("/[1-3] of 3/", text)>0;
run;
```

The data acrf is displayed in output 1.

```
Obs text
1   1 of 3
2   Form: Date of Visit
3   2 of 3
4   Form: Informed Consent
5   3 of 3
6   Form: Demographics
```

Output 1. The Output from Get_form.sas

Alternatively, Python's re module can be used to perform this task.

The program is get_form.py.

```
#get_form.py
import re
errors = []
linenum = 0

pattern = re.compile(r"of 3", re.IGNORECASE)
pattern1 = re.compile(r"Form:", re.IGNORECASE)

outfile=open('formname.txt','w')

with open ('acrf.txt', 'rt') as myfile:
    for line in myfile:
        linenum += 1
        if pattern.search(line) != None:
            errors.append((linenum, line))
        if pattern1.search(line) != None:
            errors.append((linenum, line))

for err in errors:
    print("Line " + str(err[0]) + ": " + err[1], file=outfile)
```

The output 2 shows the output from get_form.py.

```
Line 3: 1 of 3

Line 4: Form: Date of Visit

Line 7: 2 of 3

Line 8: Form: Informed Consent

Line 15: 3 of 3

Line 16: Form: Demographics
```

Output 2. The Output from Get_form.py

STEP 2

Now the aCRF.pdf is to be saved as acrf.fdf. Open the aCRF.pdf, click on Tools, comment. All the comments are shown. Click on the … , choose export all to data file and save it as fdf file acrf.fdf.

Display 2 shows a paragraph from the acrf.fdf when it is opened with notepad.

```
4 0 obj
<</C[1.0 1.0 0.0]/Contents(SV.SVSTDTC)/CreationDate(D:20200713231037-
04'00')/DA(0.898 0.1333 0.2157 rg /Helv 12 Tf)/DS(font: Helvetica,sans-serif
12.0pt; text-align:left; color:#E52237 )/F 4/M(D:20200713231441-
04'00')/NM(72a9a471-8c5b-424f-b02f-2f4ac9cab1fa)/Page 1/RC(<?xml
version="1.0"?><body xmlns="http://www.w3.org/1999/xhtml"
xmlns:xfa="http://www.xfa.org/schema/xfa-data/1.0/"
xfa:APIVersion="Acrobat:20.9.0" xfa:spec="2.0.2"  style="font-
size:12.0pt;text-align:left;color:#FF0000;font-weight:normal;font-
style:norma\
l;font-family:Helvetica,sans-serif;font-stretch:normal"><p dir="ltr"><span
style="font-family:Helvetica">SV.SVSTDTC</span></p></body>)/Rect[314.359
638.234 422.359 656.634]/Subj(Text Box)/Subtype/FreeText/T(hliu)/Type/Annot>>
endobj
```

Display 2. A Sample Paragraph from acrf.fdf

In this sample paragraph the strings of interest are "Contents(SV.SVSTDTC)/Creation" and "/Page 1/RC" as they contain the annotation and the page number. The SAS function PRXMATCH is used to find those strings. After the annotation is identified, strings of two patterns are extracted from the annotation: DOMAIN.VARIABLE and DOMAIN.QNAM=. This is achieved through SAS functions PRXPARSE, PRXSUBSTR and PRXNEXT. The program is read_fdf.sas.

```
** read_fdf.sas ** ;

data afdf;
infile "/xxxx/xxxx/acrf.fdf" print lrecl=2000 pad missover;
input text $char2000.;
run;


data match(keep=anno page_num); set afdf;
pos1=prxmatch("/\/Contents\(/", text);
pos2=prxmatch("/\)\/Creation/", text);
anno=substrn(text, pos1+10, pos2-pos1-10);

pos3=prxmatch("/\/Page /", text);
pos4=prxmatch("/\/RC/", text);
page_num=substrn(text, pos3+1, pos4-pos3-1);

if anno>' ';
run;


data match; set match;
retain pat1 pat2;
if _N_=1 then do;
pat1=prxparse("/\w+\.\w+/");
pat2=prxparse("/\w+\.QNAM=\w+/");
end;

call prxsubstr(pat2, anno, start, length);
```

```
if start gt 0 then do;
match0 = substrn(anno,start,length );
end;

start=1;
stop=length(anno);
call prxnext(pat1, start, stop, anno, position, length);

array match[3] $20.;
do i = 1 to 3 while (position gt 0);
match[i] = substr(anno,position,length);
call prxnext(pat1,start,stop,anno,position,length);
end;
run;
```

PROC REPORT is used to display the data MATCH in the output 3.

page_num	anno	match0	match1	match2	match3
Page 0	STUDYID=XXXXX-XXXX-001				
Page 1	SV=Subject Visits				
Page 1	SV.SVSTDTC		SV.SVSTDTC		
Page 1	SV.SVENDTC		SV.SVENDTC		
Page 2	DS=Disposition				
Page 2	DM=Demographics				
Page 2	SUPPDS=Supplemental Qualifier for DS				
Page 2	DM.RFICDTC		DM.RFICDTC		
Page 2	DS.DSSTDTC where DS.DSDECOD='INFORMED CONSENT OBTAINED'		DS.DSSTDTC	DS.DSDECOD	
Page 2	SUPPDS.QVAL where SUPPDS.QNAM=TIVER	SUPPDS.QNAM =TIVER	SUPPDS.QVAL	SUPPDS.QNAM	
Page 3	DM=Demographics				
Page 3	SUPPDM=Supplemental Qualifiers for DM				
Page 3	DM.BRTHDTC		DM.BRTHDTC		
Page 3	DM.SEX		DM.SEX		
Page 3	DM.RACE		DM.RACE		
Page 3	SUPPDM.QVAL where SUPPDM.QNAM=RACEOTH	SUPPDM.QNAM =RACEOTH	SUPPDM.QVAL	SUPPDM.QNAM	

Output 3. The Sample Output from Read_fdf.sas

Alternatively, Python's re module can be used to perform this task. In Python there is the function re.findall that can find all the matches of a regular expression. The program is read_fdf.py.

```
#read_fdf.py
import re

errors = []

linenum = 0

pattern = re.compile(r"\/Page [0-9]*\/", re.IGNORECASE)

pattern1 = re.compile(r"Contents", re.IGNORECASE)


outfile=open('outfile.txt','w')


with open ('acrf.fdf', 'rt') as myfile:
    for line in myfile:
        linenum += 1
        if pattern.search(line) != None:
            result=line.index('Page')
            result2=line.index('/RC')
            result3=line[result:result2]
            errors.append((linenum, result3))

        if pattern1.search(line) != None:
            result5=re.search(r"Contents", line).start()
            result6=re.search(r"\)\/Creation", line).start()
            result7=line[result5+9:result6]

            errors.append((linenum, result7))

            pattern2=re.findall(r"[\w]+[\.][\w]+",result7)
            pattern3=re.findall(r"[\w]+[\.]QNAM=[\w]+", result7)

            for pat2 in pattern2:
                 errors.append((linenum, pat2))
            for pat3 in pattern3:
                 errors.append((linenum, pat3))

for err in errors:
    print("Line " + str(err[0]) + ": " + err[1], file=outfile)
```

The output 4 shows the sample output from the read_fdf.py.

```
Line 7: Page 0
Line 7: STUDYID=XXXXX-XXXXX-001
Line 11: Page 1
Line 11: SV=Subject Visits
Line 15: Page 1
Line 15: SV.SVSTDTC
Line 15: SV.SVSTDTC
Line 19: Page 1
Line 19: SV.SVENDTC
Line 19: SV.SVENDTC
Line 23: Page 2
Line 23: DS=Disposition
Line 27: Page 2
Line 27: DM=Demographics
Line 31: Page 2
Line 31: SUPPDS=Supplemental Qualifier for DS
Line 35: Page 2
Line 35: DM.RFICDTC
Line 35: DM.RFICDTC
Line 39: Page 2
Line 39: DS.DSSTDTC where DS.DSDECOD='INFORMED CONSENT OBTAINED'
Line 39: DS.DSSTDTC
Line 39: DS.DSDECOD
Line 43: Page 2
Line 43: SUPPDS.QVAL where SUPPDS.QNAM=TIVER
Line 43: SUPPDS.QVAL
Line 43: SUPPDS.QNAM
Line 43: SUPPDS.QNAM=TIVER
```

Output 4. Sample Output from Read_fdf.py

STEP 3

PROC TRANSPOSE can be used to transpose the data created in Step 1 or the data created by Python in Step 2. Page number is used as key variable to merge the data from the Step 1 and Step 2. The SAS program and output for Step 3 are not provided as this step is straightforward.

In the merged dataset there are the following variables: page number, form name, annotations and SDTM variable names including the value of QNAM.

Such a dataset can be generated for each version of aCRF of the same study. Comparison of those datasets shows the change of annotations in each form.

From this dataset programmer can collect the SDTM domain names, variable names and the aCRF page numbers. This information can be used for the tasks below:

- This can be compared with the SDTM specifications. Programmer can detect if an SDTM variable is in the aCRF but not in the SDTM specifications.
- This can be compared with the metadata extracted from SDTM datasets. Programmer can detect if an SDTM variable is in the aCRF but not in the SDTM datasets.
- This can be used to add CRF page numbers to the origin of variable in SDTM data specifications, which is quite efficient in comparison with entering CRF page numbers manually into the data specifications. The CRF page numbers are required when the programmers create the define.xml.

CONCLUSION

SAS has an arsenal of functions to handle regular expressions. These functions can be used to analyze complex text files and extract important information. Python's re module can perform the same task.

Either SAS or Python programming can create from aCRF a dataset that has the variables for page number, form name and annotations. This dataset is used to compare different versions of aCRF and check the aCRF against SDTM data specifications and SDTM datasets.

REFERENCES

Cody, Ron. 2004. "An Introduction to Perl Regular Expressions in SAS 9". *Proceedings of SUGI 29, Paper 265-29.*

CONTACT INFORMATION

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

Hengwei Liu
Daiichi Sankyo, Inc.
211 Mount Airy Road
Basking Ridge, NJ 07920
Hengwei_liu@yahoo.com

APPENDIX

The three-page aCRF used in this paper is displayed in the appendix.

Case Report Forms

Study XXXXX-XXXXX-001

STUDYID=XXXXX-XXXXX-001

Form: Date of Visit SV=Subject Visits

Date of Visit		DD MMM YYYY
	SV.SVSTDTC	SV.SVENDTC

SUPPDS=Supplemental Qualifier for DS

DS=Disposition

DM=Demographics

Form: Informed Consent

Date Subject or Legally authorized person signed the informed consent	DD MMM YYYY
DS.DSSTDTC where DS.DSDECOD='INFORMED CONSENT OBTAINED'	DM.RFICDTC
Protocol Version Consented under	1.0 ◯
	2.0 ◯
SUPPDS.QVAL where SUPPDS.QNAM=TIVER	3.0 ◯
	4.0 ◯

DM=Demographics

SUPPDM=Supplemental Qualifiers for DM

Form: Demographics

Date of Birth

DD MMM YYYY

DM.BRTHDTC

Sex

DM.SEX

Male

Female

Race

DM.RACE

White

Black or African American

Asian

American Indian or Alaska Native

Native Hawaiian or Other Pacific islander

Other

Race other, specify

SUPPDM.QVAL where SUPPDM.QNAM=RACEOTH

3 of 3