PharmaSUG 2017 - Paper AD22

A novel method to track mapping of all CRF variables into SDTM datasets

Aishhwaryapriya Elamathivadivambigai, Seattle Genetics, Inc., Bothell, WA

ABSTRACT

A common approach to ensure mapping of all CRF variables into SDTM datasets is by manually comparing annotated CRF pages with mapping documents. However, this approach is highly time consuming, error prone and more tedious, especially in situations when new variables are added to CRFs as the study progresses. Hence, a user-friendly automated approach or a technique that could facilitate evaluating whether all CRF variables are successfully mapped to SDTM domains is highly desirable. So, in an attempt to provide a solution, this paper introduces a method that can automatically evaluate whether or not all CRF variables are properly mapped to SDTM datasets and also report a list of CRF variables that are yet to be mapped to the user. This approach accurately captures all the CRF variables from the CRF document and compares each variable with the SDTM program log files. The programming techniques used for this approach and a working example will be described in this paper.

INTRODUCTION

A case report form (CRF) is a data collection tool in clinical trials which accurately represents the data to be obtained in a protocol of a clinical trial. The patient information is appropriately captured in each page of a CRF. The variables captured in the CRF are mapped into SDTM (Study Data Tabulation Model) datasets. This is crucial because all patient information has to be entered into SDTM datasets so they will be available when they are required in the analysis.

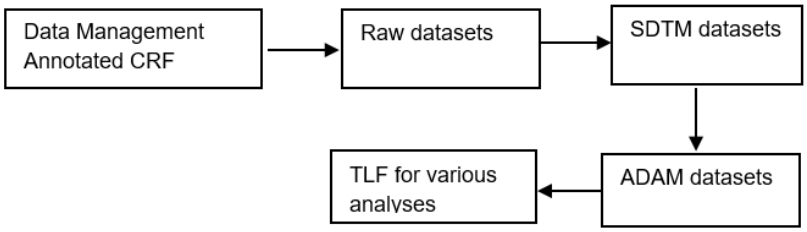

Figure 1: Flow of clinical trial process from eCRF to TLF (Table, Listing, Figure) generation

Figure 1 explains the work flow of a typical clinical trial process, right from the availability of eCRFs to TLF generation. The data collected in each CRF page are captured in an appropriate raw dataset by the Data Management team, which inturn are used for SDTM datasets generation. The CRFs annotated by the Data Management team are available as eCRFs for the programmers' review. As a next step, the aforementioned work flow eCRFs are further annotated by programmers to accurately denote how each variable would be mapped to appropriate SDTM domain(s). As new data may be captured as the study progresses it may become increasingly diffcult to check if all the variables have been mapped into SDTM datasets. So, this paper attempts to propose a technique to evaluate whether or not all CRF variables are mapped into SDTM datasets.

Enrollment/Screening [ENROLL]

Screening ID SCREENID (num, 8)	<system-generated, see note below>
Do not enter unless patient has signed informed consent.	
Patient Initials SUBJINIT (char, 3)	
Patient ID PTID (char, 8)	<system-generated, see note below>

Figure 2 Sample eCRF (Data Management team annotated CRF)

Figure 2 shows the Sample Data management annotated CRF. The variables highlighted in red represent the eCRF variables.

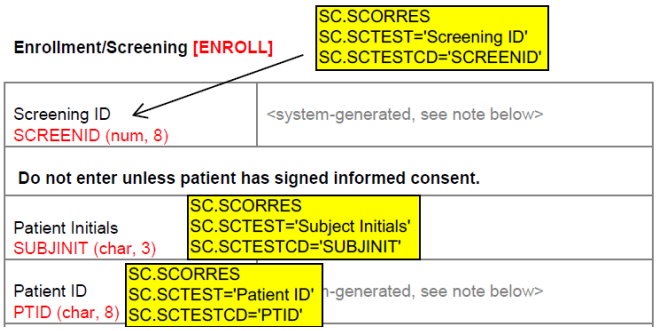

Figure 3 Programmer annotated CRF

Figure 3 shows how the CRF variables would be mapped into SDTM datasets. With the programmer annonated CRFs in hand, SDTM domains are developed by adhering to SDTM guidelines.

METHODOLOGY

CAPTURING OF ALL ECRF VARIABLES IN A SAS® DATASET

1. The eCRFs annotated by the data management were available in .pdf format. eCRFs were first converted from .pdf to .txt format and later imported using PROC IMPORT procedure in SAS to produce a dataset that contained entire information available in an eCRF distributed as variables.

```
proc import datafile="C:\path\05.txt" out=mydata dbms=dlm replace;
   getnames=no;
run;
```

2. Each *mydata* variable was concatenated to a single variable in a pre-processing step. The regex pattern search was adopted to look for actual CRF variable names in the variable obtained by concatenation.

3. A regex pattern was built to recognize and capture CRF variables. The regex below means: a word boundary followed by two or more capital letters, then follwed by another word boundary. A word boundary denotes the start or the end of a string of letters, digits and/or underscores.

```
if _N_ = 1 then pattern_num = prxparse("/\b[A-Z,_,0-9]{2,}\b/");
retain pattern_num;
```

   The following piece of code was executed to capture both the starting position and the length of the CRF variables. Thus the string matching the regex patter was detected and extracted.

```
call prxsubstr(pattern_num, var, mypos, mylength);
if mypos ^= 0 then word = substr(var, mypos, mylength);
```

   This pattern recognition algorithm captures any word that starts with an uppercase alphabet, has two or more characters or has a number or an underscore (e.g.: LABEL, LABEL1, AETERM etc.). Apart from capturing CRF variable names, it also captures information that may not be CRF variable names. The corrections implemented to counter this are presented in the following steps.

4. The sample eCRF used has a specific fomat to define each page name; pagemanes were enclosed within square brackets as shown in Figure 2. (Eg: Enrollment/Screening [ENROLL]). The following code snippet shows how each eCRF page name was capturded.

```
if find(var,'[')>0 and find(var,']')>0 then do;
   a=find(var,'[');
   b=find(var,']',a+1);
```

```
  crfpage1=substr(var,a+1,b-1);
  if find(crfpage1,']')=length(crfpage1) then   substr(crfpage1,length(crfpage1),1)=' ';
  if _N_ = 1 then pattern_num = prxparse("/\b[A-Z]{2,}\b/");
  retain pattern_num;
  call prxsubstr(pattern_num, crfpage1, mypos, mylength);
  if mypos ^= 0 then crfpage = substr(crfpage1, mypos, mylength);
end;
```

VAR	LABEL	PAGE
ABNDESC	If cytogenetics have not normalized, describe	LBMRD
ABNSPEC	If Cytogenetic Type is Abnormal, describe	DXDIAG
ACU	Acute Care Unit	LRMRU
ACU_RAW	Acute Care Unit (Character)	LRMRU
ADDSPEC	If another reason not listed above applies, please specify	INTTHER
ADVAGE	Advanced Age	INTTHER
ADVAGE_RAW	Advanced Age (Character)	INTTHER
AE	Related to an Adverse Event that started during the treatment period and is ongoing	LRMRU
AEENDTC	Condition End Date (DD/MMM/YYYY)	AE
AEENDTC_DD	Condition End Date (DD/MMM/YYYY) Day	AE
AEENDTC_INT	Condition End Date (DD/MMM/YYYY) Interpolated	AE
AEENDTC_MM	Condition End Date (DD/MMM/YYYY) Month	AE
AEENDTC_RAW	Condition End Date (DD/MMM/YYYY) (Character)	AE
AEENDTC_YYYY	Condition End Date (DD/MMM/YYYY) Year	AE
AEOUT	Outcome of event	AE
AEOUT_STD	Outcome of event Coded Value	AE
AEREL	Was this related to Blinded Study Treatment?	AE
AEREL2	Was this related to HMA?	AE
AEREL2_STD	Was this related to HMA? Coded Value	AE
AEREL_STD	Was this related to Blinded Study Treatment? Coded Value	AE

Figure 4 The dataset *fromcrf* with eCRF page names and variable names captured from eCRF document

Figure 4 shows the dataset *fromcrf* that was created in Step 4. The variables shown above do not always represent the eCRF variables, as all upcased words from the eCRF are captured in the step 4. The variables var and page represents the word extracted from the eCRF and each page name in the eCRF respectively.

5. After each eCRF page name along with CRF variables were succesfully captured in a SAS dataset *fromcrf*, all the variables from the entire collection of raw datasets were captured to another dataset called *contents*. This was achieved through the below attached piece of code.

```
proc contents data=raw._all_ noprint out=contents;
run;
```

VAR	LABEL	PAGE
ABNDESC	If cytogenetics have not normalized, describe	LBMRD
ABNSPEC	If Cytogenetic Type is Abnormal, describe	DXDIAG
ACU	Acute Care Unit	LRMRU
ACU_RAW	Acute Care Unit (Character)	LRMRU
ADDSPEC	If another reason not listed above applies, please specify	INTTHER
ADVAGE	Advanced Age	INTTHER
ADVAGE_RAW	Advanced Age (Character)	INTTHER
AE	Related to an Adverse Event that started during the treatment period and is ongoing	LRMRU
AEENDTC	Condition End Date (DD/MMM/YYYY)	AE
AEENDTC_DD	Condition End Date (DD/MMM/YYYY) Day	AE
AEENDTC_INT	Condition End Date (DD/MMM/YYYY) Interpolated	AE
AEENDTC_MM	Condition End Date (DD/MMM/YYYY) Month	AE
AEENDTC_RAW	Condition End Date (DD/MMM/YYYY) (Character)	AE
AEENDTC_YYYY	Condition End Date (DD/MMM/YYYY) Year	AE
AEOUT	Outcome of event	AE
AEOUT_STD	Outcome of event Coded Value	AE
AEREL	Was this related to Blinded Study Treatment?	AE
AEREL2	Was this related to HMA?	AE
AEREL2_STD	Was this related to HMA? Coded Value	AE
AEREL_STD	Was this related to Blinded Study Treatment? Coded Value	AE

Figure 5 Proc contents output

Figure 5 shows the proc contents output, this procedure captures all the variables, labels and page names from each raw dataset. So, this dataset contains all the system generated variables (e.g.: AEENDTC_DD, AEENDTC_MM etc.) apart from eCRF variables.

6. The two datasets *fromcrf* and *contents* were appropriately merged by data page name and variable name to produce a dataset containing all the CRF page names and their eCRF variables; thus eliminating all the system generated variables which were originally absent in the eCRF pages but present in the raw datasets. This step also eliminated the unnecessary variables in the dataset *fromcrf* which were not part of the raw dataset and thus the resultant dataset *crfvar* contained only the variables present in both *fromcrf* and *contents* which were only eCRF variables.

PAGE	LABEL	NAME
R.AE	Condition End Date (DD/MMM/YYYY)	AEENDTC
R.AE	Outcome of event	AEOUT
R.AE	Was this related to Blinded Study Treatment?	AEREL
R.AE	Was this related to HMA?	AEREL2
R.AE	AE #	AESEQ
R.AE	Was this serious?	AESER
R.AE	NCI Common Toxicity Grade	AESEV
R.AE	Onset date (DD/MMM/YYYY)	AESTDTC
R.AE	Condition/Event Description	AETERM
R.AE	Is the condition related to a protocol procedure? (Response required if event started after Informed Consent and before 1st Dose)	RELATED
R.AE	Onset Period	START
R.AE	Onset time relative to HMA	STRTTIM1
R.AE	Onset time relative to Blinded Study Treatment	STRTTIME
R.AEYN	Has the patient experienced an Adverse Event or Pre-existing condition?	AEYN

Figure 6 Dataset with all the CRF variables and page names

Figure 6 shows the dataset *crfvar* that contains only the eCRF variables. Name represent the variables name and page represents the page name.

CONVERTING SDTM LOG FILES INTO A SAS DATASET AND PRE-PROCESSING

7. SDTM log files are text file which contain the entire SAS code along with notes, warning and comments. As the next step SDTM log files were imported to SAS using the following piece of code.

```
proc import datafile="C:\path\ae.log" out=sdtm1 dbms=dlm  replace;
            getnames=no;
run;
```

8. The *sdtm1* dataset obtained from the above procedure contained many variables, then all variables were concatenated to a single variable to perform pre-processing. By concatenating all the variables into a single variable the entire SAS log was converted into a single variable in the dataset *sdtm1*. This facilitated the removal of all unnecessary segments in *sdtm1*.

9. The dataset created in step 8 were cleaned to eliminate unwanted information captured in the log files. Comments in SAS can be in any formats. E.g.: /*This is a comment*/ or *This is comment; or comment This is a comment. The following code snippet was used to accomplish this task.

```
/*deleting all the comments in the sdtm log*/
  slash1=find(var,'/*')>0; slash2= find(var,'*/');len=length(var);
  if slash2=(len-1) and slash1>0 then delete;
  if slash1=1 and slash2=0 then delete;
  if slash2=(len-1) and slash1=0 then delete;

  mid1=find(var,'/*');mid2=find(var,'*/');
  if mid1>0 and mid2>0 and length(var)-1 ne mid2 then
     del=scan(var,1,'/*')||' '||scan(var,3,'*/');
  if mid1 > 0 and mid2=0 then add1=scan(var,1,'/*');
  if mid2 > 0 and mid1=0 then add2=scan(var,2,'*/');
```

```
slash11=find(var,'*')>0; slash21= find(var,';');len1=length(var);
if slash21=(len1-1) and slash11>0 then delete;
slash12=find(var,'comment')>0; slash22= find(var,';');len2=length(var);
if slash22=(len2-1) and slash1>0 then delete;
```

VAR
294 DM LOG CLEAR;
295 PROC DATASETS LIB = WORK KILL NOLIST;
296 RUN;
297 QUIT;
300 PROC SORT DATA = R.AE OUT = AE_ (DROP = STUDYID SITEID AESTDTC AEENDTC AESER) TAGSORT;
301 BY SUBJECT;
302 RUN;
303 PROC SORT DATA = R.AEYN OUT = AEYN_ (DROP = STUDYID SITEID) TAGSORT;
304 BY SUBJECT;
305 RUN;
306 PROC SORT DATA = R.EOS OUT = EOS_ (DROP = STUDYID SITEID AESPID RENAME=(SUBJECT=SUBJID)) TAGSORT;
307 BY SUBJECT;
308 RUN;
309 PROC SORT DATA = R.EX OUT = EX1 (DROP = STUDYID SITEID) TAGSORT;
310 BY SUBJECT;
311 RUN;
312 PROC SORT DATA = R.EXA OUT = EXA1 (DROP = STUDYID SITEID) TAGSORT;
313 BY SUBJECT;
314 RUN;

Figure 7 SAS log converted into a dataset

10. Figure 7 shows the SAS code, where R.AE, R.AEYN etc were used as the input datasets. R represents the library where the raw datasets were stored. As the raw dataset names and the eCRF page names were exactly the same; the raw dataset/CRF page names were extracted from the dataset *sdtm1* by looking up for the string 'r.X', where X is the raw dataset name. These were stored in the dataset *raw*.

```
if find(var,'r.','i')>0;
```

VAR
r.ae
r.aeyn
r.eos
r.ex
r.exa
r.exd

Figure 8 The raw datasets names/eCRF page names extracted from the dataset sdtm1

Figure 8 shows the raw dataset names which were used in the SAS code for the SDTM program log file which were imported in Step 7. This shows us that the eCRF variables from these raw datasets/eCRF pages were used in the program. The variables var was renamed as page to facilitate the next step.

11. The datasets *crfvar* and *raw* were merged by the page to produce a resulting dataset called *final* with the CRF variables. These variables were searched across the *sdtm1* dataset to search for an exact match; which represented the usage of the variables in the SDTM program.

PAGE	LABEL	NAME
R.AE	Condition End Date (DD/MMM/YYYY)	AEENDTC
R.AE	Outcome of event	AEOUT
R.AE	Was this related to Blinded Study Treatment?	AEREL
R.AE	Was this related to HMA?	AEREL2
R.AE	AE #	AESEQ
R.AE	Was this serious?	AESER
R.AE	NCI Common Toxicity Grade	AESEV
R.EOS	If Death, Primary Cause of Death (AE#)	AESPID
R.EX	Primary AE# causing dose delay	AESPIDD
R.EXA	Primary AE# causing dose delay	AESPIDD
R.EXD	Primary AE# causing dose delay	AESPIDD
R.EX	Primary AE# causing unplanned dose adjustment	AESPIDDA
R.EXA	Primary AE# causing unplanned dose adjustment	AESPIDDA
R.EXD	Primary AE# causing unplanned dose adjustment	AESPIDDA

Figure 9 The dataset final

Figure 9 shows the dataset *final* where variable name represents the CRF variables to check in the SDTM (eg: AE) program log.

CAPTURING OF ALL CRF VARIABLES ASSOCIATED TO SDTM DOMAINS

12. Each CRF variable name captured in the dataset *final* was extracted and stored in a separate macro variable, (one macro variable for each variable name), in the following fashion: Name1 = CRF variable name1, Name2 = CRF variable name2, Name3 = CRF variable name3, etc. This was done to allow search for each of these variables in *sdtm1* dataset one by one using a find function.

13. The following code snippet was used to evaluate if all the CRF variable names, captured in separate macro variables, were present in the dataset *sdtm1*. Here, the macro variable count represents the number of macro variables generated in Step 12.

```
%do k=1 %to &count;
var1=find(var,"&&name&k ");
 if var1 > 0 then  variable1="&&name&k";
%end;
```

Here, the macro variable count represents the number of macro variables generated in Step 12.

VARIABLE
AEENDTC
AEOUT
AEREL
AEREL2
AESEQ
AESER
AESEV
AESPID
AESTDTC
AETERM
AEYN
DEATHDTC
EXSTDTC
EXSTTM
RELATED
START
STRTTIM1
STRTTIME

Figure 10 eCRF variables used in AE (SDTM) dataset

Figure 10 shows the eCRF variables that were used to derive SDTM variables in AE dataset.

14. Steps 1 through 13 were run in an iterative do loop, with each loop evaluating different SDTM domains.
15. The variables used in all SDTM domains collected in Step 14 were compared with the eCRF variables produced in *crfvar* to generate a list of CRF variables that were not used in any SDTM domains.

CONCLUSION

Since it is more likely that much new information may be captured in CRF documents as the study progresses, it is highly recommended that we adopt a technique or a method to evaluate whether or not all variables are mapped at SDTM level. With the proposed technique, all the unmapped CRF variables can be easily captured, thus averting subsequent delays in output generation.

ACKNOWLEDGMENTS

I take this opportunity to thank my supervisors Kiran Cherukuri and Jay Gadhiya for their timely suggestions and advices. I also would like to extend my sincere thanks to Rajeev Karanam, the Clinical Programming Team Director for reviewing the paper.

CONTACT INFORMATION

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

Name : Aishhwaryapriya Elamathivadivambigai
Enterprise : Seattle Genetics, Inc.
Address : 21823 - 30th Drive S.E. Bothell, WA 98021
Work Phone : 425-527-2668
E-mail : aelamathivadivambiga@seagen.com