PharmaSUG 2018 - Paper DS-12

An Alternate Way to Create the Standard SDTM Domains

Sunil Kumar Pusarla, Omeros Corporation

ABSTRACT

Sponsors who initiate clinical trials after 2016-12-17 are required to submit data from the studies in CDISC Study Data Tabulation Model (SDTM) format in their FDA submission. SDTM Implementation Guides provide the rules, assumptions, and examples necessary to create the standard domains. It is common practice to use software tools to assess compliance with CDISC standards before transmitting the domains to a regulatory agency. By pre-building the standard domain templates directly from the CDISC Share data stores, compliance with metadata (type, storage, and internal documentation) is guaranteed. These version-specific templates are then used as the scaffolding to which study-specific data are added. This paper outlines how this top-down approach minimizes coding issues as well as errors/warnings raised by metadata checking tools. A macro with dual functionality will be presented. One function utilizes the metadata library to produce specific standard domains variables, labels and their canonical order in that domain. The second function derives the maximum length of each individual character variable and assigns that length to that variable. This paper is intended for people new to CDISC requirements and operating with limited corporate resources.

INTRODUCTION

FDA has mandated that, for NDAs, BLAs, ANDAs and their subsequent submissions, the clinical data must be submitted in CDISC formats for studies that start after 2016-12-17. This paper is intended for an audience with limited corporate resources, and provides insight into building a metadata (provided by CDISC share) driven solution to implement the SDTM standards. Sponsors with abundant resources, can afford to develop their own or purchase third party MDRs (Metadata Directory Repository) to maintain the SDTM IG standards; hence it is easy for them to implement the regulatory standards. For sponsors with limited resources, it is difficult to develop or obtain third-party MDRs.

It is common practice to verify the SDTM domains using tools such as Pinnacle 21. Based on generated warnings/errors, one makes the necessary changes to SAS code to address them wherever applicable, and if not, explain them in a Reviewer's Guide. By relying on CDISC published data standards, and a single, streamlined macro, one avoids the aggravation of tracking down and adjusting myriad miniscule code details across multiple domains.

OVERVIEW

Rather than creating the SDTM domain, and then relying on Pinnacle 21 tool to verify compliance, one can use CDISC share provided SDTM metadata as a foundation, minimizing the problems created and subsequent code modifications.

We created a SAS library *(Figure 1)* from the CDISC share metadata. It has all the individual standard SDTM domains. Each domain *(i.e. Figure 2)* has its variables in the required order. The variables are associated with their respected labels. By using these templates, one reduces the use of many ATTRIB or LABEL statements to assign the labels to each variable *(Figure 3*) during their creation. This approach also avoids typing (spelling or casing) errors. One should use only LENGTH statement while creating the character standard variables. Because the order is provided in these templates, it assures the final dataset variables will be in the required order.

FDA requires the EPOCH variable in most of these domains, and each character variable must match the maximum length of its longest observed length. CDISC SDTM share metadata doesn't provide EPOCH variable in all the domains. We included the EPOCH variable in each standard domain where required, and at the required position within the created SAS library *(Figure 1)*. Our macro assigns the appropriate labels to the standard SDTM variables, calculates the maximum observed length of each variable, and then orders the variables to comply with the specifications.

Figure 1 – SDTM standard domains SAS library

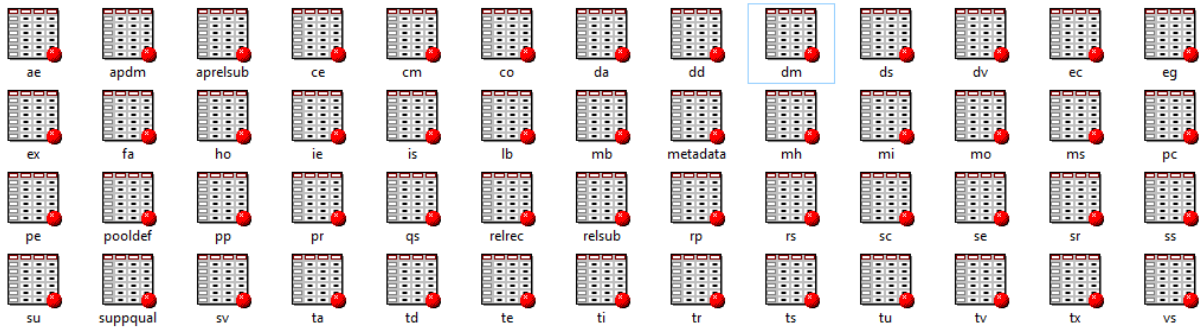

Figure 2 – SDTM Demographics (DM) domain

STUDYID	DOMAIN	USUBJID	SUBJID	RFSTDTC
1	2	3	4	5

RFENDTC	RFXSTDTC	RFXENDTC	RFICDTC	RFPENDTC
6	7	8	9	10

DTHDTC	DTHFL	SITEID	INVID	INVNAM
11	12	13	14	15

BRTHDTC	AGE	AGEU	SEX	RACE
16	17	18	19	20

ETHNIC	ARMCD	ARM	ACTARMCD	ACTARM
21	22	23	24	25

COUNTRY	DMDTC	DMDY
26	27	28

Figure 3 – Assigning labels and lengths – the tedious way

The end process consists of three steps: building the penultimate dataset; calling the macro, which generates two global macro variables, and then creating the desired SDTM domain in a regular Data step, which employs the above macro variables.

The penultimate dataset is created using a combination of CDISC guidelines (variable names) and local specifications (defining age, Study Day #).

Then, one calls the macro by supplying at least two keyword parameters. The first (*ind*) specifies your penultimate dataset name to modify (apply variable labels); the second (*supply*) specifies the template dataset name that supplies the CDISC share metadata labels and appropriate variable order. Two global macro variables are produced during macro execution.

These will be used in a very simple Data step to create the final domain. Macro variable *&varlist_length* contains the string of required variables with their derived lengths (delimited by a space) in the required order, to be used with length statement in the final Data step. Macro variable *&varlist* is a list of the mandatory variables to keep in that standard domain.

CONCLUSION

While creating SDTM domains, developing a small set of macro tools enables one to achieve compliance with regulatory guidelines with minimal effort and minimizes the chance of error. It ensures adherence to the data standards and reduces the possibility of creating metadata errors and having to fix them in post processing.

The opinions expressed in this paper are the opinions of the author and should not be interpreted as the opinion or position of Omeros Corporation.

REFERENCES

https://www.cdisc.org/

http://documentation.sas.com/?docsetId=mcrolref&docsetTarget=titlepage.htm&docsetVersion=9.4&locale=en

ACKNOWLEDGMENTS

My sincere thanks to our director Mr. Paul Hamilton for encouraging, suggesting and helping in writing this paper.

CONTACT INFORMATION

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

Name: Sunil Kumar Pusarla
Enterprise: Omeros Corporation
Address: 201 Elliott Avenue West
City, State ZIP: Seattle, WA 98119
Work Phone: 206-676-5000
E-mail: skumar@omeros.com

Macro and its usage are presented below:

```
32
33 %macro LablLen(ind = , supply =, dlib = );
34   %local dat varnum sqvar vnum ;
35   %global varlist_length varlist;
36
37   %let dat = flengs1;
38   %* Applying appropriate labels to your data variables ;
39
40   %if %length(&ind) eq 0 %then %do;
41     %let ind = &syslast;
42   %end;
43
44   %if %length(&dlib) eq 0 %then %do;
45     %let dlib = cdisc;
46   %end;
47
48   proc contents data = &dlib..&supply
49                 out = cdisc_&supply. (keep = name label varnum)
50                 nodetails
51                 noprint;
52   run;
53
54   proc sort data = cdisc_&supply. ;
55     by name ;
56   run;
57
58   proc contents data = &ind.
59                  out = yourdata (keep = name varnum type)
60                        noprint ;
61   run;
62
63   data yourdata;
64     set yourdata;
65     name = upcase(name);
66   run;
```

```
 67
 68  proc sort data = yourdata ;
 69    by name;
 70  run;
 71
 72  data ourdata;
 73    merge yourdata       (in = a rename = (varnum = vnum))
 74          cdisc_&supply. (in = b);
 75    by name;
 76    if a and b;
 77    length varlab sqvar $200 ;
 78    varlab = catx(' = ',name,quote(strip(label)));
 79    if (type eq 2) then sqvar = catx('',
 80                                     cats('max(length(',name,'))'),
 81                                     'as',
 82                                     cats('len',varnum)
 83                                     );
 84     else if (type eq 1) then sqvar = cats('8 as len',
 85                                           varnum);
 86  run;
 87
 88  proc sql noprint;
 89    select varlab, sqvar, varnum into :varlab separated by ' ',
 90                                      :sqvar  separated by ',',
 91                                      :varnum separated by '/'
 92      from ourdata
 93    order by varnum;
 94  quit;
 95
 96  proc datasets library = work nolist;
 97    modify &ind;
 98    label &varlab ;
 99    run;
100  quit;
101
```

```
102   %* Get the max length of your data variables ;
103
104   proc sql ;
105     create table lengs as
106     select &sqvar.
107     from &ind;
108   quit;
109
110   proc transpose data = lengs
111                   out = tlengs;
112     var len:;
113   run;
114
115   proc sql ;
116     create table flengs as
117       select a.*, b.col1 as lengt
118       from ourdata a left join tlengs b
119     on strip(scan(a.sqvar,-1,' ')) eq b._name_;
120   quit;
121
122   data &dat;
123     set flengs;
124     length wanted $10;
125     if (type eq 2) then wanted = cats('$',lengt);
126     else                wanted = strip(put(lengt,8.));
127     if  (substr(name,length(name) - 1) eq 'FL')
128                               then wanted  = '$1';
129     if length(name) ge 6 then do;
130     else if (substr(name,length(name) - 3) eq 'STAT')   then wanted  = '$8';
131     end;
132   run;
133
```

```
134   proc sql noprint;
135     select catx(' ', upcase(name),wanted),
136            name,
137            varnum
138                   into: varlist_length separated by ' ',
139                       : varlist separated by ' ',
140                       : vnum separated by ' '
141     from &dat
142     order by varnum;
143   quit;
144 %mend;
```

Example Demographics (DM) dataset creation:

```
data dm0;
  set rawdata.dm (rename = (race = race1) drop = studyid siteid);
  length usubjid race raceoth1 raceoth2 $40;
  studyid = project;
  domain = 'DM';
  subjid = subject;
  siteid = sitenumber;
  usubjid = catx('/',
                 studyid,
                 subjid);
  rficdtc = put(input(compress(consdt_raw),date9.),yymmdd10.);
  if index(brthdat_raw,'UN') and
     index(brthdat_raw,'JUN') lt 1 then put 'WAR' 'NING: Fix the below.';
  brthdtc = put(input(compress(brthdat_raw),date9.),yymmdd10.);
  ageu = 'YEARS';
  ethnic = upcase(ethnic);
  race = race_std;
  if (race eq 'OTHER') then do;
    put /race1;
    raceoth1 = scan(race1,1,'/');
    raceoth2 = scan(race1,2,'/');
    if index("&pgmpath",'prod') then put 'WAR' 'NING: Check the race other.';
  end;
  sex = first(sex);
  visit    = 'VISIT 1';
  visitnum = 1;
run;
```

… (Continue to create the penultimate dataset and then use the macro.)

```
data all1;
  set all;
  by usubjid;
  if (input(rfstdtc,yymmdd10.) ne .) and
     (input(dmdtc,yymmdd10.) ne .) then
     dmdy = input(dmdtc,yymmdd10.) - input(rfstdtc,yymmdd10.) ;
  if (dmdy ge 0) then dmdy = dmdy + 1;
run;

%lablLen(ind = all1, supply = dm)

options varlenchk = nowarn;
data sdtm.dm (label = "Demographics" keep = &varlist);
  length &varlist_length;
  set all1;
run;
options varlenchk = warn;

%makexpt(sdtm.dm)
```