PharmaSUG China2017 - 59

Create Zero Observation data set to achieve maximum metadata resolution in CDISC Submission

M K Sinha (Ajay), Incedo Technology Solutions Limited, Bangalore, India

ABSTRACT

In a world of systematic and efficient programming the success of project delivery largely depends on the accuracy of data and implementation of approved standards. The metadata across all the files (analysis/mapping specification file, raw data and final data) need to match for standardization and traceability. CDISC standards for all most all of the data structures clearly states metadata information of a variable, its order of appearance in a dataset, variable length etc. In order to achieve the same metadata as is specified by company or CDISC standard files, programmers need to be very careful while generating the data set at the final stages. Most of time we enter into situations where number of variables are more, or the order or metadata is different from one actually required in the final data set.

Also since these data sets (say SDTM, ADaM) in most cases are developed from SAS coding after referring the analysis/mapping specification file, end moment change in the specification file may not be carried over to these data sets. One approach to limit this error and generate the deliverables per standards is to first develop the analysis/mapping speciation file as per guidelines and then import this file into SAS to create a metadata from these analysis/mapping file. Use these metadata to assign the metadata for final data set. This paper elaborates on the technique of first creating a zero observation data set with the metadata as same as specified in the analysis/mapping specification spreadsheet and then to use this data set to create the final data set. In this approach the analysis/mapping specification file needs to be current and the resulting data sets will be generated as per standard. A review of final data set will double check the metadata consistency not only in the resulting data sets but also on the analysis/mapping specification file.

INTRODUCTION

Development of specification files predominantly is the statistician work, and in order to achieve result (data set) in desired format statistician also should pay attention from the initiation of project. Use CDISC (Clinical Data Interchange Standard Consortium) specification file available online and customize it as per requirement or develop a customized specification file per requirement. Import this spreadsheet in SAS and generate a zero observation data having same metadata as specified in specification file and later club it with the programmer's final data set to generate the standardized data set. In the productive environment where the challenge is to generate data set following algorithms, doing complex analysis if metadata consistency is somehow automated will be a great help for the programmers. By developing a macro which can take care of metadata consistency of final dataset programmers are set free to concentrate only on programming to generate quality output. This method not only eliminates the rework which a programmer has to do every time there is any update to specification file but also create a solid link between specification file and data set. With the current specification file that is used to generate the standardize dataset, generation of define.xml also becomes scalable.

CREATION OF SAS IMPORT-TABLE SPREADSHEET

Use a 97-2003 format spread sheet as this is very stable platform to rely on. Online available CDISC datasets metadata files can be used and the information can be copied in the former spread sheet. Use any process available in SAS to import the information in a form of SAS dataset. A proper parity can be maintained between the variables that need to be displayed in the final datasets and variables for which presence is uncertain. Several checks can be introduced to capture only the variables which a statistician is comfortable with. Some variables which are uncertain can be controlled not to be carried over to final dataset by these checks. (One such check can be carrying only those variables from spreadsheet which are in UPPER CASE). It is also important to set some bench mark rule or rule for SAS log error when any metadata is not consistent with specification file. e.g. introduce a check in the SAS macro which throws error in SAS log whenever the length for any variable is specified more than 40. These checks can validate the data/metadata even before final data set is generated which in turn improves the quality of data generated and eliminates last minute hustle due to these issues.

Display 1. is an example of Specification Spreadsheet.

Label	Subject-Level Analysis Dataset
Key Variables	STUDYID USUBJID
Dataset Structure	One record per subject
Author	M K SINHA (AJAY)
Note	

ADaM Standard								
Variable Name	Variable Label	Variable Type	Length	Notes or Mapp	Origin	Mapped Variable	Codelist / Controlled Terms	Role
STUDYID	Study Identifier	Char	12		DM.STUDYID		(STUDYID)	IDENTIFIER
SITEID	Study Site Identifier	Char	4		DM.SITEID			IDENTIFIER
USUBJID	Unique Subject Identifier	Char	21		DM.USUBJID			IDENTIFIER
BIRTHDT	Birth Date (N)	Num	8		DM.BRTHDTC			TIMING
CONSDTC	Consent Date	Char	11		DS.CONSDTC			TIMING
CONSDTN	Consent Date (N)	Num	8		CONSDTC			TIMING
RANDDTC	Randomization Date	Char	11		RD.RANDDTC			TIMING
RANDDTN	Randomization Date (N)	Num	8		RANDDTC			TIMING
TR01SDT	Date of First Exposure in Run-In	Num	8		EX.EXDTC			TIMING
TR01STM	Time of First Exposure in Run-In	Num	8		EX.EXSTMC			TIMING
TR01SDTM	Datetime of First Exposure in Run-In	Num	8		TR01SDT, TRT01STM			TIMING
TR01EDT	Date of Last Exposure in Run-In	Num	8		EX.EXDTC			TIMING
TR01ETM	Time of Last Exposure in Run-In	Num	8		EX.EXSTMC			TIMING
TR01EDTM	Datetime of Last Exposure in Run-In	Num	8		TR01EDT, TRT01ETM			TIMING

Display 1. Snapshot of Specification Spreadsheet

IMPORT SPECIFICATION SPREADSHEET

Various data models developed by CDISC like SDTM and ADaM can be developed using the mapping specification or analysis specification spreadsheet. SAS has different procedures to import these spreadsheet and the easiest approach will be to use “PROC IMPORT”

Sample Code:

```
** call required tab data from excel specification **;

    %macro tabcall(tab=);
        PROC IMPORT OUT= work.&tab
            DATAFILE= "&spec_loc"
            DBMS=EXCEL REPLACE;
            SHEET="&tab$";
             GETNAMES = NO;
        RUN;
    %mend tabcall;
```

Display 2. is an example of SAS Imported Specification Spreadsheet.

	F1	F2	F3	F4	F5	F6
1	Label	Subject-Level Analysis Dataset		.		
2	Key Variables	STUDYID USUBJID		.		
3	Dataset Structure	One record per subject		.		
4	Author	M K SINHA (AJAY)		.		
5	Note			.		
6	ADaM Standard			.		
7	Variable Name	Variable Label	Variable Type	.	Notes on Mapping	Origin
8	STUDYID	Study Identifier	Char	12		DM.STUDYID
9	SITEID	Study Site Identifier	Char	4		DM.SITEID
10	USUBJID	Unique Subject Identifier	Char	21		DM.USUBJID
11	BIRTHDT	Birth Date (N)	Num	8		DM.BRTHDTC
12	CONSDTC	Consent Date	Char	11		DS.CONSDTC
13	CONSDTN	Consent Date (N)	Num	8		CONSDTC
14	RANDDTC	Randomization Date	Char	11		RD.RANDDTC
15	RANDDTN	Randomization Date (N)	Num	8		RANDDTC
16	TR01SDT	Date of First Exposure in Run-In	Num	8		EX.EXDTC
17	TR01STM	Time of First Exposure in Run-In	Num	8		EX.EXSTMC
18	TR01SDTM	Datetime of First Exposure in Run-In	Num	8		TR01SDT, TRT01STM
19	TR01EDT	Date of Last Exposure in Run-In	Num	8		EX.EXDTC
20	TR01ETM	Time of Last Exposure in Run-In	Num	8		EX.EXSTMC
21	TR01EDTM	Datetime of Last Exposure in Run-In	Num	8		TR01EDT, TRT01ETM
22						

Display 2. Snapshot of Specification Spreadsheet after import in SAS

AUTO DIFFERENTIATE BETWEEN SUBJECT LEVEL DATASET AND BASIC DATA STRUCTURE

One of the major difference between subject level analysis dataset (ADSL) and other analysis datasets is the presence of common/core variables in other dataset. Create an auto call to differentiate between ADSL and Basic Data Structure (BDS).

Sample Code:

```
%if &dsn ne adsl %then %do;
        ** -- Call analysis dataset tab -- **;
        %tabcall (tab = &&dsn);
        ** -- Call Common Variables tab  -- **;
        %tabcall (tab = COMMON_VARIABLES);
    %end;
    %else %do;
        ** -- Call analysis dataset tab -- **;
        %tabcall (tab = &&dsn);
    %end;
```

MACRO VARIABLE TO STORE DATASET LABEL, AND NAME

To make the process automated it's always good to use the resources available in SAS itself. One of the efficient way is to use SAS dictionary or sashelp library. These are real time resources which gets updated depending on SAS usage. Use the view files in sashelp (like vcolumn, vtable etc) to extract real data. Create a global macro variable to store the name and label of data set that can be used later depending on usage.

Sample Code:

```
**------------------------------------------------------------------------------------------------**
                    ANALYSIS DATASETS METADATA CREATION
**------------------------------------------------------------------------------------------------**;
  ** Create a macro variable to store analysis dataset label, and name **;
  proc sql noprint;
        select f2 into: dsnlabel
                from &dsn_
                where seq = 1;
                %let dsnname = &dsn_;
                %put &dsnname;
                %put &dsnlabel;
  quit;
```

Display 3. is an example of macro variable capturing data set name and label.

```
MLOGIC(MZOBS):  %LET (variable name is DSN_)
MLOGIC(MZOBS):  %PUT &dsn_
adsl
MLOGIC(MZOBS):  %LET (variable name is DSNNAME)
MLOGIC(MZOBS):  %PUT &dsnname
adsl
MLOGIC(MZOBS):  %PUT &dsnlabel
Subject-Level Analysis Dataset
NOTE: PROCEDURE SQL used (Total process time):
      real time               0.00 seconds
      cpu time                0.00 seconds
```

Display 3. Snapshot of SAS Log capturing resolved values of data set name and label

VARIABLE TO HOLD METADATA ATTRIBUTES

Once the import of spreadsheet and storage of data set name and label into a macro variable is complete the next target is to create a mask of dataset with the metadata as is defined in the spreadsheet. The variable need to be defined with the same metadata as present in spreadsheet.

Sample Code:

```
**create variable to hold the attributes and metadata **;
data &dsn;
  set &dsn(where =(upcase(f1)= f1));
   length var $200;
   &dsnname = strip(&dsnname);
   if f1 ne '';
   var1 = compbl(strip(f1)||''||strip(f3)||'('||strip(put(f4,8.))||') '|| 'label='||strip('"'||f2));
   var = strip(strip(var1)||'"');
   if strip(upcase(f3)) = 'CHAR' and upcase(strip(f8)) = 'ISO8601' then var2  = strip(var)||' format =
$'||strip(put(f4,best.))||'.'|| ' informat = $'||strip(put(f4,best.))||'.';
   if var2 ne '' then var = var2;
run;
```

CREATE MACRO VARIABLE THAT HOLDS ALL VARIABLE NAME

There is a requirement from programming perspective to keep only the variables that are present in the spreadsheet in final data set. In the heat of programming often this is missed out. So it's a good approach here to create a macro variable which holds all the variable name sequentially as per spreadsheet, which can be used later depending on requirement.

Sample Code:

```
proc sql;
        select f1 into : var separated by " " from &dsn_;
quit;

%put &var;
** create Zero Observation dataset **;
proc sql noprint;
        select var into :var1  separated by ' , ' from &dsn_ ;
quit;
```

Display 4. is an example of a macro variable capturing all the variables in sequential order.

```
MLOGIC(MZOBS):  %PUT &var
STUDYID SITEID USUBJID BIRTHDT CONSDTC CONSDTN RANDDTC RANDDTN TR01SDT TR01STM TR01SDTM TR01EDT
TR01ETM TR01EDTM
NOTE: PROCEDURE SQL used (Total process time):
      real time           0.00 seconds
      cpu time            0.00 seconds
```

Display 4. Snapshot of SAS Log capturing resolved values of macro variable Var - All the variables name in same order as in spreadsheet

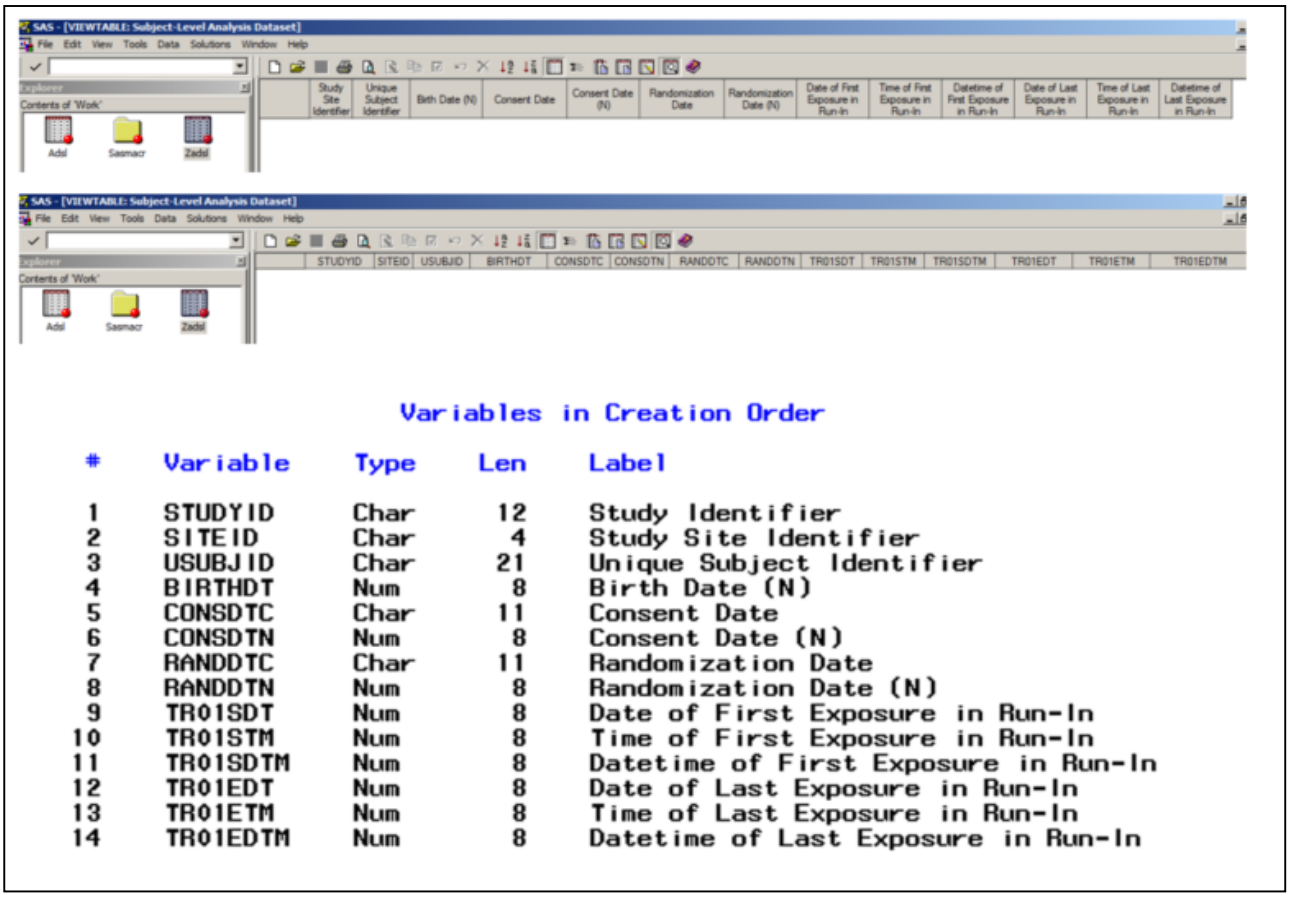
CREATE ZERO OBSERVATION DATSET

Once the above mentioned requirement is full filled the ultimate step will be to create a zero observation data set. This data set will not have any observation, but only the metadata as defined in the imported spreadsheet. Once the final programming is completed by a SAS programmer this zero observation data set can be merged with the final data set in such a way that the resultant data set captures the metadata from zero observation data set and data from the final data set. In this way the resultant data set will be as per spreadsheet requirement, eliminating various programming tasks. This will result in increasing the quality of data with programmer concentrating on the derivation, analysis etc and not on metadata.

Sample Code:

```
proc sql;
  create table z&dsn (label = "&dsnlabel") (&var1);
quit;
```

Display 5. is an example of zero observation data set.



Variables in Creation Order

#	Variable	Type	Len	Label
1	STUDYID	Char	12	Study Identifier
2	SITEID	Char	4	Study Site Identifier
3	USUBJID	Char	21	Unique Subject Identifier
4	BIRTHDT	Num	8	Birth Date (N)
5	CONSDTC	Char	11	Consent Date
6	CONSDTN	Num	8	Consent Date (N)
7	RANDDTC	Char	11	Randomization Date
8	RANDDTN	Num	8	Randomization Date (N)
9	TR01SDT	Num	8	Date of First Exposure in Run-In
10	TR01STM	Num	8	Time of First Exposure in Run-In
11	TR01SDTM	Num	8	Datetime of First Exposure in Run-In
12	TR01EDT	Num	8	Date of Last Exposure in Run-In
13	TR01ETM	Num	8	Time of Last Exposure in Run-In
14	TR01EDTM	Num	8	Datetime of Last Exposure in Run-In

Display 5. Snapshot of Zero Observation SAS Data set with metadata as defined in spreadsheet

CREATION OF COMMON VARIABLES

For BDS data there is requirement of common variable. In this regard the imported Common variable tab from the specification spreadsheet can be a good source to create a macro variable. These common variables can be later added along with core variable for that dataset to generate a zero observation dataset that will have common and core variable together which can be masked with the dataset generated by a programmer to form final dataset as per spreadsheet.

Sample Code:

```
**------------------------------------------------------------------------------------------------------------------------**
                                    COMMON VARIABLES CREATION
**------------------------------------------------------------------------------------------------------------------------**;
        data cmnvar ;
                set COMMON_VARIABLES;
                if upcase(f1) = f1;
        run;
        proc sql;
                select f1 into : cmnvar separated by " " from cmnvar;
        quit;
        %put &cmnvar;
```

Display 6. is an example of common variable tab of spreadsheet.

Label	Common Varaibles				
Key Variable	STUDYID USUBJID				
Dataset Struc	One record per subject				
Author	M K SINHA (AJAY)				
Note					
ADaM Standard	ADaM Standard				
Variable Name	Variable Label	Variable Type	Length	Codelist / Controlled Terms	Source / Derivation
STUDYID	Study Identifier	Char	12		ADSL
SITEID	Study Site Identifier	Char	4		ADSL
USUBJID	Unique Subject Identifier	Char	19		ADSL
SUBJID	Subject Identifier for the Study	Char	6		ADSL
CONSDTC	Consent Date	Char	10		ADSL
CONSDTN	Consent Date (N)	Num	8		ADSL
TRTSTDT	Date of Fisrt Dose	Num	8		ADSL
TRTTM	Time of First Dose	Num	8		ADSL
TRTSTDTM	Datetime of First Dose	Num	8		ADSL
AGE	Age	Num	8		ADSL
AGEU	Age Units	Char	10		ADSL
SEX	Sex	Char	1		ADSL
RACE	Race	Char	40		ADSL
RACEOTH	Race - Other	Char	200		ADSL
ETHNIC	Ethnicity	Char	40		ADSL
ENRLFL	Enrolled Population Flag	Char	1		ADSL
SAFFL	Safety Population Flag	Char	1		ADSL
MITTFL	Modified Intent-to-Treat Population Flag	Char	1		ADSL
EFFFL	Efficacy Population Flag	Char	1		ADSL
PPROTFL	Per-Protocol Population Flag	Char	1		ADSL
TRT01	Treatment	Char	40		ADSL
TRT01N	Treatment (N)	Num	8		ADSL
HEIGHT	Enrollment Height (cm)	Num	8		ADSL
WEIGHT	Enrollment Weight (kg)	Num	8		ADSL
DSCOMP	Completed Protocol	Char	1		ADSL
DSVISNUM	Number of last visit completed	Char	40		ADSL
DSREASON	Reason for Early Study Termination	Char	200		
DOSENUM	Number of Doses Administered	Num	8		

Display 6. Snapshot of Common Variable Spreadsheet for BDS

Display 7. macro variable capturing all the common variables in sequential order.

```
MLOGIC(MZOBS):  %PUT &cmnvar
STUDYID SITEID USUBJID SUBJID CONSDTC CONSDTN TRTSTDT TRTTM TRTSTDTM AGE AGEU SEX RACE RACEOTH
ETHNIC ENRLFL SAFFL MITTFL EFFFL PPROTFL TRT01 TRT01N HEIGHT WEIGHT DSCOMP DSVISNUM DSREASON
DOSENUM
```

Display 7. Snapshot of SAS Log - Global Macro Variable CMNVAR resolved to capture all common variables

USAGE IN MAIN DATASET SAS PROGRAM

Develop an automatic approach to define your libraries (using SAS admin). Create a macro variable which captures the name of SAS program, let's call this macro variable as dsn. The tab on specification spreadsheet has the same name as of resultant data set and so dsn macro variable can be triggered in such a way that it automatically imports the required tab into SAS once the data set SAS program is run for it. Later through the macro create a zero observation data set, so if this macro is run on adsl , then ZADSL will be created in work library with all the metadata as per spreadsheet. Later merge this ZADSL with the dataset created during programming to achieve the requirement.

Sample Code:

```
%zobs(dsn = "&dsn");

**Your SAS Program to create final data set let's call the final generated data set by a programmer as
FINAL**;

** create data set as per spreadsheet requirement**;
data derived.adsl(label="&dsnlabel" keep = &var);
retain &var;
```

```
            set z&dsn final;
    run;
```

CONCLUSION

In the programming world SAS programmers often miss the variable metadata as is required for submission. There are chances that some variables are either missed or not populated well in the data set. This can be captured during dual programming, but if there is an easy way out and if this can be eliminated at first instance itself then programmers can concentrate on the quality of data rather than metadata. Though this approach seems easy various other methods can be clubbed with it to make it robust. This can be later extended to run all the data set programs together and generate the resultant ouput.

CONTACT INFORMATION <HEADING 1>

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

Name: M K SINHA (AJAY)
Enterprise: INCEDO TECHNOLOGY SOLUTIONS LIMITED
Address: BANGALORE, INDIA
City,StateZIP: BANGALORE , KARNATAKA 560103
Work Phone: +91-9886161421/ 80-6708 5878
Fax:+91 80 6708 5839
E-mail:mk.sinha@incedoinc.com, mksinhacr@gmail.com
Web:https://www.incedoinc.com/