PharmaSUG 2013 - Paper DS07

Title: Macro %D_ADSL – Automating ADSL Creation from Metadata File

Author: JIANHUA (DANIEL) HUANG – Celgene Corporation

Key words:

ADaM, Macro

Abstract:

As CDISC standards are being more broadly accepted in clinical trials, many tools have been developed for automating SDTM creation. Conversely, there are not many tools available for creating ADaM datasets due to two reasons: First, the ADaM data structures are more flexible than SDTM; and second, the ADaM derivations are more complex and study specific. Both reasons make it difficult to handle ADaM derivations within one macro program. Fortunately, we can define the ADaM structure and algorithm in the metadata, and then use the metadata to create ADaM datasets. A macro program (called %D_ADSL) has been developed to automate ADSL creation by reading the information from its metadata file. The macro first creates each variable in a pre-specified order, from a simple copy of the SDTM variable to the variables with most complicated derivation. Then it reads all variable attributes (i.e. name, label, length, and controlled terms) from the metadata into ADSL. In addition to creating the ADSL dataset, the macro can also output SAS® code for self review and for further modification. Because this macro is using the metadata to create ADSL, it overcomes the challenge of automatically creating ADSL while handling the complexity of derivation. More importantly, the macro improves the traceability of ADSL derivations and ensures the consistency between ADSL and the metadata.

Outline of this article:

1. Review ADSL subject-level data structure

ADSL is also called the subject-level dataset because it has a one subject, one record structure. It contains study core variables such as: study identifiers, population flags, planned and actual treatment, demographic information, stratification and sub-grouping variables, important trial dates, etc. ADSL also contains other subject-level variables that are important in describing a subject's experience in the trial. This unique one subject, one record structure of ADSL shall be identical across all clinical trials, regardless of the type of clinical trial design. In addition, ADSL and its related metadata are required in a CDISC-based submission from a clinical trial even if no other analysis datasets are submitted.

2. Review ADSL metadata file and the rational of automating ADSL creation

2.1. Introduction of ADSL metadata file.

The ADaM metadata facilitates the communication for the underlying assumptions, statistical methods, transformations, derivations and imputations performed in the analysis of a clinical trial. It provides "the information about data". There are four types of metadata, among them, the variable metadata define the detailed algorithm and attributes for analysis variables. Below table 1 and table 2 are examples of ADSL variable metadata:

Table 1, ADSL variable metadata.

Variable	Label	Type	Format	Codelist	Comments
STUDYID	Study Identifier	Text	$5		DM: STUDYID
USUBJID	Unique Subject Identifier	Text	$25		DM: USUBJID
SITEID	Study Site Identifier	Text	$10		DM: SITEID
SUBJID	Subject Identifier for the Study	Text	$10		DM: SUBJID
AGE	Age	integer	8		DM: AGE
AGEGR1	Pooled Age Group 1	Text	$5	< 65 >=65	If . < AGE < 65, then AGEGR1 ="< 65" Else if AGE >= 65, then AGEGR1 =">= 65"
AGEGR1N	Pooled Age Group 1 (N)	integer	8	AGEGRP65	Numeric values for AGEGR1
ITTFL	Intent-To-Treat Population Flag	Text	1	Y,N	SUPPDM file; Use QVAL where QNAM="ITT". Refer to SAP Section 3.2.

The variable metadata of ADSL defines all variables' attributes (i.e. variable name, label, type, and format), and their source or derivation algorithm (under 'Comments'). As you can see from 'Comments' column, some variables are simply copied from SDTM data, such as: STUDYID, USUBJID, SITEID, SUBJID, AGE while others need to be derived, i.e. AGEGR1 is derived from AGE with algorithm defined in 'Comments' column. Among those derived variables, some might need more complicated derivations, such as 'ITTFL', while others might be generated from a variable codelist, such as 'AGEGR1N'.

Table 2, variable codelist.

CODELST	DATATYPE	CODEVAL	RNK	DECOD
TEST	C	R		RED
TEST	C	Y		YELLOW
AGEGRP65	N	1		< 65
AGEGRP65	N	2		>= 65
AGEGRP75	N	1		< 75
AGEGRP75	N	2		>= 75
SEX	N	1		F
SEX	N	2		M
RACE	N	1		ASIAN
RACE	N	2		BLACK OR AFRICAN AMERICAN
RACE	N	3		NATIVE HAWAIIAN OR OTHER PACIFIC ISLANDER
RACE	N	4		NORTH AMERICAN INDIAN OR ALASKA NATIVE
RACE	N	5		WHITE, NON-HISPANIC AND NON-LATINO
RACE	N	6		WHITE, HISPANIC OR LATINO
RACE	N	7		OTHER

The variable codelist should be included in metadata file as well. It defines how the grouping and sub-grouping variables are coded as well as their decode values. If the codelist was already defined in SDTM data, then the same codelist should be copied into the ADaM metadata. Otherwise, the codelist should be created based upon study documents (i.e. SAP, CRF annotation) and ADaM guidelines. It is recommended that same codelists are used across all studies for the same drug compound.

2.2. The rationale of automating ADSL creation.

The unique subject-level data structure of ADSL supports the rationale for automatically generating ADSL datasets through a macro program. The metadata file provides the information sources of ADSL derivation. Based on the complexity of the derivation we can classify ADSL variables into different groups, and then derive each group of variables in a hierarchical order. The variables with the simplest derivation algorithm will be created first. We then will use these variables to create more complicated variables and repeat the process until the most complicated variables have been derived. Once completed, we assign all variables' attributes from copying the same information in metadata file. A detailed illustration of creating ADSL variables from %D_ADSL macro is giving in below paragraphs.

3. Introduction of macro %D_ADSL

%D_ADSL is a SAS® macro program that generates ADSL datasets by reading information from its metadata file.The five major steps are described below:

3.1. Step 1- Import ADSL metadata and codelist. For editing purpose, the metadata is usually saved in an Excel file. Following SAS® code demonstrates how to read in ADSL metadata and codelist from Excel file into SAS® dataset. “&infile” is a macro parameter that indicates the location and file name of the metadata. An example of &infile will be: *infile=%str(S:\xxxx\DDT\metadata_training.xls).*

```
libname ddt excel "&infile"  mixed=yes;

*-read in code list;
 data codelist;
    set ddt.'CODELIST$'n;
 run;

*-read in ADSL variable metadata;
 data  adslddt;
    set ddt.'ADSL$'n;
 run;

libname ddt clear;
```

3.2 Step 2 - Create SAS® format/informat from metadata codelist. Following SAS® code indicates how to convert the codelist into SAS® format.

```
data  format(keep=fmtname start end label type);
  set codelist;
 *-Create formats from code list;
              fmtname=strip(codelst)||'F';
              start=put(codeval, $60.);
              end=put(codeval, $100.);
              type=datatype;
              label=put(decod, $100.);
              output;
*-Create Informats from code list;
             fmtname=strip(codelst)||'F';
             start=put(decod, $100.);
             end=put(decod, $100.);
             if datatype='N' then type='I';
             else if datatype='C' then type='J';
             label=put(codeval, $100.);
             output;
run;

*-Bring in SAS format/informats;
 proc format CNTLIN=format;
 run;
```

3.3 Step3 - Derive ADSL variables by sequence.

As mentioned before, ADSL variables will be classified into different groups and then be derived in a hierarchical order. To achieve this step we first add the following new columns into the ADSL metadata excel file: "*Source, Sequence, Macro, Code, Intervar*" (see table 3 below), then assign a sequence number for each variable according to the complexity of the derivation. Variables with 'Sequence=1' will be directly copied from 'Source' SDTM data and will be created first. Variables with 'Sequence=2' will be generated based on 'Sequence=1' variables with some simple derivation. The derivation SAS® code is given under 'Code' column. Variables with 'Sequence=3' will be derived based on sequence 1 & 2 variables. These can be created in several ways: either by calling a standard macro program, by applying a SAS® format defined in a codelist, or by using more complicated SAS® code specified in 'Code' column. We can assign additional sequence numbers and continue the loops until the most complicated variables are created. Sometimes, in order to create a higher hierarchy variable, an intermediate variable is needed, but this variable is not necessary to be kept in the final ADSL dataset. This variable will be specified as "Intervar=Y" and added in the metadata Excel file as a new record. The intermediate variable is not only a bridge to create more advanced variables but also improves derivation traceability. Finally, when the loop is ended, if any variables are still unable to be generated by the macro, we assign them as 'Sequence=99' and they will be coded manually in SAS®.

Table 3, ADSL metadata file with new columns added.

Variable	**Source**	**Sequence**	**Macro**	**Code**	**Intervar**
STUDYID	SDTM.DM	1			
USUBJID	SDTM.DM	1			
AGE	SDTM.DM	1			
AGEGR1N	AGE	2		if.z<age<65then agegr1n=1;else if age>=65 then agegr1n=2;	Y
AGEGR1	AGEGR1N	3		agegr1=put(agegr1n, agegrp65f.);	
TRTSDT	SDTM.DM(keep=USUBJID RFSTDTC)	2		trtsdt=input(rfstdtc, yymmdd10.);	
TRTEDT	EXSTDT	3	%GET1OBS(dataout=TRTEDT, datain=EXSTDT, function=max);		
NUMDOSE		99			

Below is the SAS® code for deriving ADSL variables in sequence. Due to the page limitations, only SAS® code of ‘Sequence=1’ variables are displayed in this paper which are variables directly copied from the SDTM data. Each variable will be created separately and saved into an individual dataset, where the dataset is identified by its own variable name. All individual data contains two variables: its own variable and ‘usubjid’, except for data USUBJID, where it has only one variable ‘usubjid’. All datasets will be sorted by ‘usubjid’ for future merges and combination purposes.

```
data adslseq1(keep=sequence variable source callfrom);
    set adslddt(where=(sequence=1));
      by sequence;
run;

proc sql noprint;
  select count(variable)
  into :nobs
  from adslseq1;
quit;

%let nobs=&nobs;

proc sql noprint;
  select variable, source, callfrom
  into :var1 - :var&nobs notrim, :data1 - : data&nobs notrim, :call1 - : call&nobs notrim
  from adslseq1;
quit;

%do i=1 %to &nobs;
%let variable=&&var&i;
%let source=&&data&i;
%let callfrom=&&call&i;

proc sql noprint;
  create table &variable as
%if &variable=USUBJID %then %do;
   select usubjid
%end;
%else %do;
    select usubjid, &variable
 %end;
   from &source
   order by usubjid;
quit;
```

3.4 Step 4 - Combine all variables together and assign variable attributes.

Once all variables are created in hierarchical order, they will be combined together by common variables ‘usubjid’. Then, variable label, format, lengths and types will be copied from the metadata and assigned to an ADSL dataset. Below is the corresponding SAS® code for copying and assigning variable attributes.

```
proc sql noprint;
  select variable, label, type, format, length
    into :var1 - :var&nobs notrim, :label1 - :label&nobs notrim, :type1 - :type&nobs notrim,
          :fmt1 - :fmt&nobs notrim, :len1 - :len&nobs notrim
    from attrib;
quit;

*-Assign variable attributes;
data &outfile;
  ATTRIB
 %do i= 1 %to &nobs;
    %let variable=&&var&i;
   %let label=&&label&i;
   %let type=&&type&i;
   %let format=&&fmt&i;
   %let length=&&len&i;
      &variable
      label="&label"
    %if %index(&type, text)>0 %then %do;
      length=$&length
    %end;
    %else %do;
     length=&length
   %end;
      format=&format
  %end;
     ;
    MERGE
    %do i= 1 %to &nobs;
      &&var&i
    %end;
    ;
    by usubjid;
run;
```

3.5 Step 5 - Output SAS® code.

In addition to creating ADSL datasets, the macro %D_ADSL can also output the corresponding SAS® derivation code and save them into a SAS® program. As mentioned before, in the end of the macro process, if there are variables unable to be created by %D_ADSL, we can use the output SAS® code as a reference and complete the programming manually for those remaining variables.

```
*-output SAS code;
 %if %upcase(&sascode)=YES %then %do;
   data _null_;
     file file1 mod;
       put @1 "/**********************************";
       put @1 "*-derive %upcase(&variable)                  ";
       put @1 "**********************************/";
       put @1 "proc sql noprint;                            ";
       put @1 "       create table &variable as             ";
         %if &variable=USUBJID %then %do;
       put @1 "         select usubjid                      ";
          %end;
          %else %do;
              put @1 "    select usubjid, &variable         ";
          %end;
       put @1 "       from &source                          ";
       put @1 "       order by usubjid;                     ";
       put @1 "quit;                                        ";
       put @1 "                                             ";
     run;
  %end;
```

4. Discussion

There are three major advantages of using %D_ADSL macro. First, the hierarchical order defined in ADSL metadata file presents a clear relationship among all ADSL variables, and also indicates the relationship between ADSL and SDTM variables. This feature significantly improves the traceability of ADSL derivations. Second, by defining the derivation algorithm in the metadata file, it largely increases the flexibility of the %D_ADSL macro and allows it to create ADSL datasets for almost all types of study designs. Third, by copying variable attributes from the metadata file into ADSL, it ensures the consistency between the metadata and ADSL datasets. Once the variable attributes are changed in the metadata file, the %D_ADSL macro will read in new information and update the ADSL variables accordingly.

There will be challenges if we want to apply the same macro to ADaM datasets other than ADSL. As most ADaM datasets, except for ADSL, are in basic data structure (BDS). The BDS structure usually contains multiple records per subject. For example, the lab data has a per subject, per visit and per parameter structure. Therefore, if we want to derive each variable individually and then combine them there will be difficulties in the merging process, as 'usubjid' is an inadequate id variable. Another challenge is that some algorithms for non-ADSL variables could be very dynamic and complicated, therefore it may not be efficient to derive them through a SAS® macro program.

5. Conclusion

The %D_ADSL macro uses the metadata file to create ADSL datasets. This new approach could improve the traceability of ADSL derivation, increase the flexibility of macro programs, and ensure the consistency between ADSL and its metadata file. Therefore, %D_ADSL macro can be used as an efficient tool for automating ADSL creation. Using the same approach to create other ADaM datasets may require further examination due to the different data structure between ADSL and other ADaM datasets.

Contact Information:

Your comments and questions are valued and encouraged. Please contact author at:

Daniel Huang, Principle SAS Programmer

Celgene Corporation

110 Allen Road, Basking Ridge, NJ 07920

Tel: 908-860-4347

Email: jihuang@celgene.com