How to Use SDTM Definition and ADaM Specifications Documents to Facilitate SAS Programming

Yan Liu
Sanofi Pasteur

ABSTRCT
SDTM and ADaM implementation guides set strict requirements for SDTM and ADaM variable attributes, including their names, labels, types, formats etc. Without a good programming solution, this can make clinical SAS programming time consuming and prone to mistakes. This paper introduces a simple method that utilizes SDTM definition and ADaM specifications documents to improve clinical SAS programming efficiency and accuracy.

KEY WORDS
SDTM, ADaM, EXCEL, MACRO, VARIABLE ATTRIBUTE, %SYSCALL SET

INTRODUCTION

The implementation of CDISC standards has greatly improved the efficiency of new drug development in the pharmaceutical industry. Common standard among the industry makes new drug application review easier for the regulators and makes global exchange smoother. In the meantime, CDISC standards make more demands for clinical SAS programming because of the strict variable attribute requirements set by the SDTM and ADaM Implementation Guides. Directly coding all the label, length and format statements in SAS for so many variables in so many data sets can be tedious and prone to mistakes.

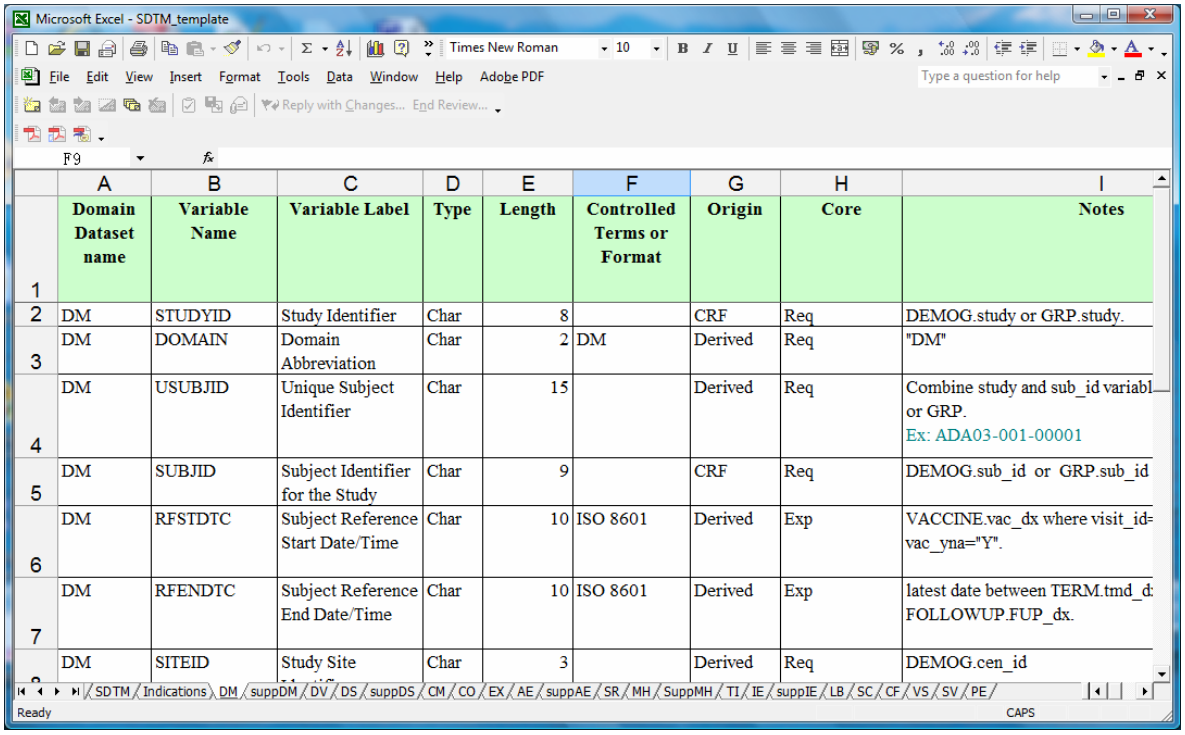
This paper introduces a simple and efficient method to meet SDTM and ADaM Implementation Guide requirements and make clinical SAS programming easier. In the following sections of this paper, I'll first introduce how to use SAS macro to convert SDTM definition and ADaM specifications documents into useful SAS data sets. Then I'll introduce how to use SAS macro to retrieve variable attributes from the data sets converted and apply them to SDTM domain or ADaM analysis data sets that will be submitted to regulators for new drug application.

CONVERT THE SDTM AND ADAM DOCUMENTS INTO SAS SYSTEM

SDTM definition and ADaM specifications documents come in Excel files. Separate

worksheets are used for each SDTM domain or ADaM analysis data set. On each of these worksheets, information about the variables for each SDTM domain and ADaM data set is listed, including the variable name, label, type, length, format, source etc. The followings are two examples of such documents.

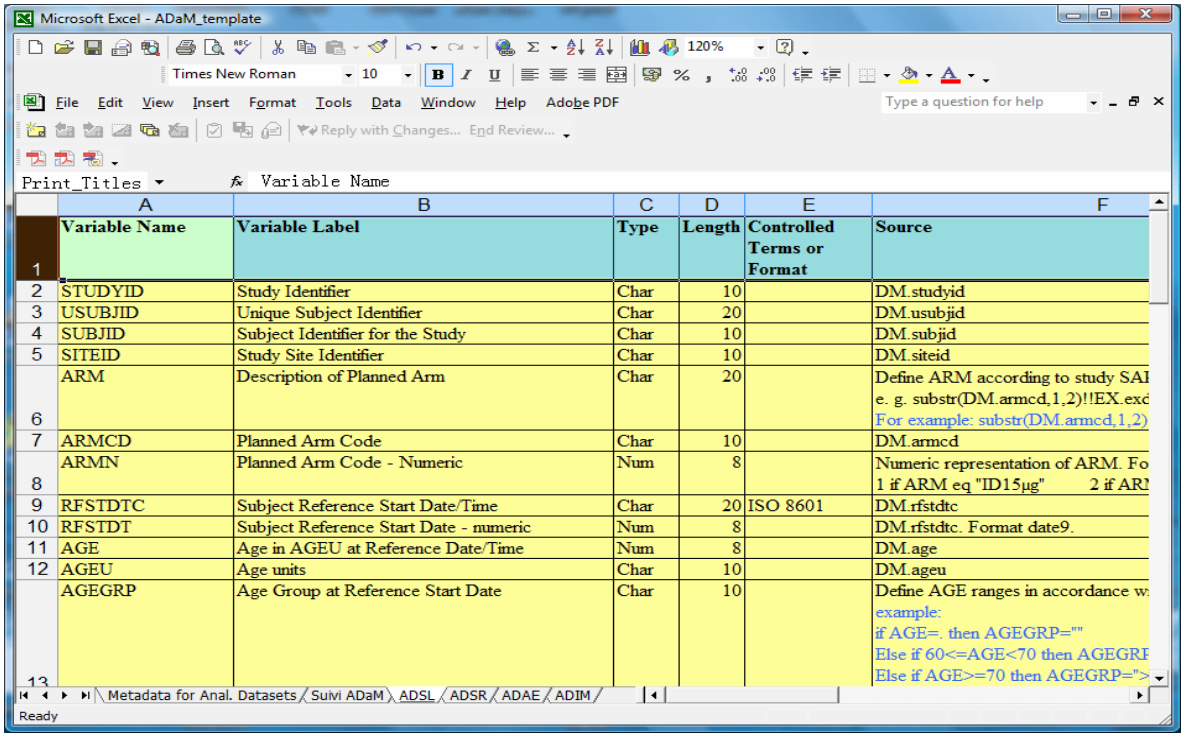
A Sample of SDTM definition document in Excel format



Microsoft Excel - SDTM_template

	Domain Dataset name	Variable Name	Variable Label	Type	Length	Controlled Terms or Format	Origin	Core	Notes
2	DM	STUDYID	Study Identifier	Char	8		CRF	Req	DEMOG.study or GRP.study.
3	DM	DOMAIN	Domain Abbreviation	Char	2	DM	Derived	Req	"DM"
4	DM	USUBJID	Unique Subject Identifier	Char	15		Derived	Req	Combine study and sub_id variabl or GRP. Ex: ADA03-001-00001
5	DM	SUBJID	Subject Identifier for the Study	Char	9		CRF	Req	DEMOG.sub_id or GRP.sub_id
6	DM	RFSTDTC	Subject Reference Start Date/Time	Char	10	ISO 8601	Derived	Exp	VACCINE.vac_dx where visit_id= vac_yna="Y".
7	DM	RFENDTC	Subject Reference End Date/Time	Char	10	ISO 8601	Derived	Exp	latest date between TERM.tmd_d: FOLLOWUP.FUP_dx.
	DM	SITEID	Study Site	Char	3		Derived	Req	DEMOG.cen_id

A Sample of ADaM specifications document in Excel format



Microsoft Excel - ADaM_template

	Variable Name	Variable Label	Type	Length	Controlled Terms or Format	Source
2	STUDYID	Study Identifier	Char	10		DM.studyid
3	USUBJID	Unique Subject Identifier	Char	20		DM.usubjid
4	SUBJID	Subject Identifier for the Study	Char	10		DM.subjid
5	SITEID	Study Site Identifier	Char	10		DM.siteid
6	ARM	Description of Planned Arm	Char	20		Define ARM according to study SAI e. g. substr(DM.armcd,1,2)!!EX.exc For example: substr(DM.armcd,1,2)
7	ARMCD	Planned Arm Code	Char	10		DM.armcd
8	ARMN	Planned Arm Code - Numeric	Num	8		Numeric representation of ARM. Fo 1 if ARM eq "ID15µg" 2 if AR
9	RFSTDTC	Subject Reference Start Date/Time	Char	20	ISO 8601	DM.rfstdtc
10	RFSTDT	Subject Reference Start Date - numeric	Num	8		DM.rfstdtc. Format date9.
11	AGE	Age in AGEU at Reference Date/Time	Num	8		DM.age
12	AGEU	Age units	Char	10		DM.ageu
13	AGEGRP	Age Group at Reference Start Date	Char	10		Define AGE ranges in accordance w example: if AGE=. then AGEGRP="" Else if 60<=AGE<70 then AGEGRI Else if AGE>=70 then AGEGRP=">"

In order to make use of these documents in SAS programming, the first thing that needs to do is to convert the information listed on these two documents into SAS data sets. The simple way to do this is to use the PROC IMPORT procedure in the SAS macro. Since the variable information for each data set is listed on separate worksheets, a SAS macro parameter should be created to select which worksheet the SAS should convert. In this paper, I name this parameter as dsnme. The first row of the excel work sheet will be kept as the data set variable names. So the GETNAMES option in the PROC IMPORT procedure should be set to Yes. If the layouts of the excel file is not as neat as the above samples and the file has extra rows on the top with extra specifications information. Then only those parts that contain variable attributes should be converted into a SAS data set. In such case, the RANGE option of the PROC IMPORT procedure should be used to specify which part of the spreadsheet SAS should convert. For example, the following statement will instruct SAS to convert only the area of the specified spreadsheet covering A12 to H50.

```
RANGE="&dsnme.$A12:H50";
```

RETRIEVE VARIABLE ATTRIBUTES AND APPLY TO SDTM DOMAIN OR ADAM ANALYSIS DATA SETS

The following SAS macro code is used to retrieve variable attributes from the temporary data set created and apply them to the SDTM domain or ADaM analysis data sets to be submitted to the regulator.

```
%let id=%sysfunc(open(_temp));
%let NOBS=%sysfunc(attrn(&id,NOBS));
%syscall set(id);

data &dsout(label="&label");
    %do i=1 %to &NOBS;
        %let rc=%sysfunc(fetchobs(&id,&i));

         %if %upcase(&type)=CHAR %then %do;
              format &Variable_Name %sysfunc(compress($&length..)) ;
              length &Variable_Name %sysfunc(compress($&length..)) ;
         %end;
         %else %if %upcase(&type)=NUM %then %do;
         %if %index(&Controlled_Terms_or_Format, .) %then %do;
             format &Variable_Name %sysfunc(compress(&Controlled_Terms_or_Format..)) ;
             length &Variable_Name %sysfunc(compress(&length..)) ;
         %end;
         %else %do;    format &Variable_Name %sysfunc(compress(&length..)); %end;
```

```
            %end;

            label &Variable_Name= "&Variable_Label";
            keep &Variable_Name;
        %end;
            set &dsin;
run;

%let id=%sysfunc(close(&id));
```

The %SYSCALL SET routine in the above SAS code creates a macro variable for each of the variable listed in the worksheet, with the macro variable names being exactly the same. The FETCHOBS function in the %DO loop retrieves variable attributes one by one and store them into the corresponding macro variables %SYSCALL SET routine created. These variable attributes stored in the macro variables then are applied through the label, length and format statements to the variables in the final data set.

Since the SAS format and length statement syntax for numeric and character variables are different, %if conditions are used. The format statement for character variables can use the value provided by the “length” column in the work sheet. But for the numeric variables, that will not be enough. If a numeric variable contains more than just integral and has decimal points, additional %if conditions are used to make sure the values in the “control terms of format” column are used as input.

The KEEP statement in the above code keeps only those variables that specified in the documents and in the same sequence as they are listed. All the other variables either temporarily created during programming or retrieved from the source data will be dropped, because they are not the needed STDM or ADaM variables. If there is a variable specified in the documents but doesn’t exist in the data set. Such variable will still be generated by the macro. But a warning of uninitialized variable will show up in the SAS log and the variable will have all null values. These will prompt the programmer to correct any forgotten variables.

A sample call of the macro looks like this,

```
%attrb(dsnme=DM, dsin=_dm, dsout=db.dm, Path=”z:\SDTM_MTA72.xls”)
```

Where, dsnme is the name of the worksheet for the data set; dsin is the data set to which you want to apply the variable attributes; dsout is the name for the final data set with the required variable attributes; Path is the location of the SDTM or ADaM documents.

CONCLUSION

Compared with coding variable attributes directly in SAS statements, this method I introduced here has many advantages. First, by utilizing the SDTM definition and ADaM specifications documents, it saves a lot of time and makes the SAS codes look more compact. Second, by using this method, I don't need to worry about whether I forgot to drop a not-needed variable or whether I forgot to create a needed variable that is specified in the documents. In addition, all the new variables will be ordered in the exact sequence as they show in the Implementation Guide. At last, since all the variables and their attributes are listed in worksheets, it is much easier to check for mistakes and make corrections once any mistakes are found. Therefore it greatly improves programming accuracy and efficiency.

REFERENCE

1. CDISC SDTM Implementation Guide V3.1.1 Final, http://www.cdisc.org/
2. the Analysis Data Model, Version 2.1, the ADaM Implementation Guide, Version 1.0, http://www.cdisc.org

AUTHOR CONTACT

Yan Liu
Sanofi Pasteur
108 Jianguo Road, Beijing, China
chris.liu@sanofipasteur.com

TRADEMARK INFORMATION