A Fast Programming Channel from DDT to Datasets in Pharmaceutical Industries

Author, Jianfeng Wang, Sanofi Pasteur, Beijing
Coauthor, Linghua Zhou, Sanofi Pasteur, Beijing

ABSTRACT

Typing the variable name, label and format from data definition table (DDT) into the SAS programs to create the corresponding SAS datasets is a time consuming task. Most of the time, the work is tedious and prone to making mistakes. The ongoing changes to the variable name, label and format also increases the difficulty of keeping them updated before the final statistical analysis. In order to deduce SAS programming work load and minimize the possibility of making errors during creating the datasets, we have explored an easy programming way which has been explained here.

In this method, the program reads the variable name, label and format from DDT which is saved as a Microsoft Excel file, it integrates the information into SAS macro variables. After that, the program exports variable name, label, format and value one by one following the sequence defined in DDT.

This method can effectively reduce the time of datasets derivation and avoid potential typing errors. With minor modification, the program can easily be applied to different datasets. If the DDT keeps being updated, we just rerun the program, and get the updated datasets. It will make our life easy and greatly improve our work's efficiency.

INTRODUCTION

When creating the Clinical Data Interchange Standards Consortium (CDSIC) submission datasets for a clinical study, one of the most time important jobs is to type the variable name, label and format from DDT line-by-line into SAS programs. Very frequently the variable name, label and format need to be changed repeatedly and even after the table outputs are generated. A minor change in the DDT will lead SAS programmers to update their programs in various locations. Mistakes could occur during this process and it will result in inaccurate final reports.

To assist SAS programmers with this tiresome work and, more importantly, to avoid potential typing errors of variable name, label and format, we developed a SAS program. Instead of manually type in the variable name, label and format from an Excel file, this program could automatically convert them into SAS macro variables and build the SAS datasets. Even more advantageously, this program can be applied to any other different domains simply after minor modification.

SECTION 1 PROCEDURE

Let's take the most common domain in clinical trials 'DM (Demographics)' as an example. There are three steps in the whole procedure, which including setting dictionary for the program, raw dataset and output 'DM' dataset; calling the variable name, label and format from DDT into SAS macro variables, and creating values of variables based on the requirements in DDT.

SECTION 1.1 SET DICTIONARY

There are many different ways to set dictionary before handling raw datasets. An example is given here which also works well.

```
%macro mypath ;
    %if   %sysfunc(getoption(dms))=dms %then %do; /*interactive mode*/
        %let pgm=%sysget(sas_execfilename);
        %let path=%sysget(sas_execfilepath);
    %end;
    %else %do; /*batch mode*/
       %let path= %sysfunc(getoption(sysin));
       %let pgm= %scan(&path, -1,"\");
    %end;
%mend;
%mypath;
```

The code above obtains the name of the program from macro variable '&pgm', 'DM.SAS' in this case, and the physical location from macro variable '&path', such as 'Z:\PROJECT NAME\PROGRAMS\DM.SAS'.

Then assign your raw dataset and output dataset to the libraries named 'db' and 'out' respectively with following code:

```
libname db "&path\..\..\datasets\raw";
options fmtsearch=(db);
libname out "&path\..\..\datasets";
```

SECTION 1.2 CALL THE VARIABLE NAME, LABEL AND FORMAT

In this core step, the program will attain the information of the variable name, label and format from the Excel document named 'sdtm.xls', which is the DDT for Study Data Tabulation Model (SDTM) datasets. The part of DM domain is showed in the below Excel table:

	A	B	C	D	E	F	G	H	I	J
1	**Domain Dataset name**	**Variable Name**	**Variable Label**	**Type**	**Length**	**Controlled Terms or Format**	**Origin**	**Role**	**Notes**	**Core**
2	DM	STUDYID	Study Identifier	Char	8		CRF	Identifier	DEMOG.study	Req
3	DM	DOMAIN	Domain Abbreviation	Char	2	DM	Derived	Identifier	"DM"	Req
4	DM	USUBJID	Unique Subject Identifier	Char	15		Derived	Identifier	Combine study and sub_id variables from DEMOG	Req
5	DM	SUBJID	Subject Identifier for the Study	Char	9		CRF	Topic	DEMOG.sub_id	Req
6	DM	RFSTDTC	Subject Reference Start Date/Time	Char	10	ISO 8601	Derived	Timing	VACCINE.vac_dx where visit_id in ('V01') and vac_yna='Y' and format to ISO8601	Exp
7	DM	RFENDTC	Subject Reference End Date/Time	Char	10	ISO 8601	Derived	Timing	TERM.trm_dx. Format to ISO8601.	Exp
8	DM	SITEID	Study Site Identifier	Char	3		Derived	Record Qualifier	00'||DEMOG.cen_id	Req
9	DM	BRTHDTC	Date/Time of Birth	Char	10	ISO 8601	CRF	Result Qualifier	DEMOG.dob_dx and format to ISO8601	Perm
10	DM	AGE	Age in AGEU at Reference Start Date/Time	Num	8		Derived	Result Qualifier	Age in Years: (RFSTDTC - DEMOG.dob_dx + 1)/365.25	Exp
11	DM	AGEU	Age Units	Char	6	YEARS, MONTHS, DAYS	Derived	Variable Qualifier	YEARS	Exp
12	DM	SEX	Sex	Char	1	M,F,U	CRF	Result Qualifier	DEMOG.sex_cl M for Male, F for Female and U for Unknown.	Req
13	DM	RACE	Race	Char	9		CRF	Result Qualifier	use the information in MTA29.DEMOG.race_cl "A" = "Asian" "B" = "Black" "C" = "Caucasian" "H" = "Hispanic" "I" = "American Indian or Alaska Native" "O" = "Other" "P" = "Native Hawaiian or other Pacific Islander"	Exp

The following program imports the Excel sheet into SAS, generates format variable which will be used in the next step and counts the number of variable names and save it into SAS macro variable '&num':

```
libname xlsdat excel "&path\..\sdtm.xls" mixed=yes;
  data xls;
  set xlsdat."dm$"n;
  where variable_name is not null;
  if type='char' then format='$'||length;else format=length;
  call symputx("num",_n_,'g');
run;
```

The following dataset is the 'xls' dataset living in the work library:

	Variable_Name	Variable_Label	Type	Length	Notes	format	
1	STUDYID	Study Identifier	Char	8	DEMOG.study	$	8
2	DOMAIN	Domain Abbreviation	Char	2	"DM"	$	2
3	USUBJID	Unique Subject Identifier	Char	15	Combine study and sub_id variables from DEMOG	$	15
4	SUBJID	Subject Identifier for the Study	Char	9	DEMOG.sub_id	$	9
5	RFSTDTC	Subject Reference Start Date/Time	Char	10	VACCINE.vac_dx where visit_id in ('V01') and vac_yna='Y' and format to ISO8601	$	10
6	RFENDTC	Subject Reference End Date/Time	Char	10	TERM.trm_dx. Format to ISO8601.	$	10
7	SITEID	Study Site Identifier	Char	3	00'||DEMOG.cen_id	$	3
8	BRTHDTC	Date/Time of Birth	Char	10	DEMOG.dob_dx and format to ISO8601	$	10
9	AGE	Age in AGEU at Reference Start Date/Time	Num	8	Age in Years: (RFSTDTC - DEMOG.dob_dx + 1)/365.25	8	
10	AGEU	Age Units	Char	6	YEARS	$	6
11	SEX	Sex	Char	1	DEMOG.sex_cl	$	1
12	RACE	Race	Char	9	DEMOG.race_cl	$	9

As you can see that the variable name, label and format are saved in varible_name, varible_label and format column respectively.

The next process is to keep the value of variable name, label and format in SAS macro variables by the code as followed:

```
proc sql noprint;
  select variable_name into:name1-:name&num
  from xls
  ;
  select variable_label into:label1-:label&num
  from xls
  ;
  select format into:format1-:format&num
  from xls
  ;
  select variable_name into:name
  separated by ' '
  from xls
  ;
quit;
```

In our example, &name1= 'STUDYID', &label1= 'Study Identifier', &format1=$8, which corresponding

to the first row of the 'xls' dataset. Then in the same order, &name2, &label2 and &format2 will keep those values in the second row of the 'xls' dataset, and so forth. The value of the macro variable '&num' keep the variable names ranged from 1 to 12 which corresponding to the second row to the thirteenth row in the Excel table for 'DM' domain and separated by blank among variable names, since the first row in the Excel table is occupied by the header in SAS dataset.

SECTION 1.3 CREATE THE VALUES OF VARIABLES.

Before moving to the next step, make sure all the information you need are included in a temporary datasets. In our case, we have to combine some raw datasets into one dataset, named 'dsn' to complete the information for DM domain, such as group number, termination date, visit date of subjects, etc.

After that, it's ready to derive the values of variables according to the definition in DDT file by the following code:

```
%macro out;
 data out.dm;
    attrib%do i=1 %to &num;&&name&i    format=&&format&i... label="&&label&i" %end;;
     set dsn;
     &name1=study;
     &name2='DM';
     &name3=trim(left(study))||'-'||trim(left(sub_id));
     &name4=sub_id;
     if dov_dx ne '' then &name5=put(input(dov_dx,date9.),yymmdd10.);
     if trm_dx ne '' then &name6=put(input(trm_dx,date9.),yymmdd10.);
     &name7='00'||trim(left(cen_id));
     if dob_dx ne '' then &name8=put(input(dob_dx,date9.),yymmdd10.);
      if dob_dx ne ''    and dov_dx ne '' then
      &name9=(input(dov_dx,date9.)-input(dob_dx,date9.)+1)/365.25;
     if &name9 ne . then &name10='YEARS ';
     &name11=sex_cl;
     &name12=race_cl;
     …
     keep &name;
  run;
%mend out;
%out;
```

The output 'DM' dataset looks like the following table:

	STUDYID	DOMAIN	USUBJID	SUBJID	RFSTDTC	RFENDTC	SITEID	BRTHDTC	AGE	AGEU	SEX	RACE	ARMCD	ARM	COUNTRY	DMDTC
1	T01	DM	T01-001-00001	001-00001	2010-11-29	2010-12-27	001	1931-07-15	79.37851	YEARS	M		1	GROUP 1	USA	2010-11-29
2	T01	DM	T01-001-00002	001-00002	2010-11-29	2010-12-20	001	1948-08-13	62.29706	YEARS	M		2	GROUP 2	USA	2010-11-29
3	T01	DM	T01-001-00003	001-00003	2010-11-29	2010-12-22	001	1941-04-16	69.62355	YEARS	M		1	GROUP 1	USA	2010-11-29
4	T01	DM	T01-001-00004	001-00004	2010-11-30	2010-12-21	001	1950-12-19	59.95072	YEARS	M		2	GROUP 2	USA	2010-11-30
5	T01	DM	T01-001-00005	001-00005	2010-11-30	2010-12-21	001	1943-02-26	67.76181	YEARS	F		2	GROUP 2	USA	2010-11-30
6	T01	DM	T01-001-00006	001-00006	2010-11-30	2010-12-22	001	1938-06-01	72.50103	YEARS	M		2	GROUP 2	USA	2010-11-30
7	T01	DM	T01-001-00007	001-00007	2010-12-01	2010-12-22	001	1948-01-08	62.89938	YEARS	M		2	GROUP 2	USA	2010-12-01
8	T01	DM	T01-001-00008	001-00008	2010-12-01	2010-12-28	001	1923-10-09	87.14853	YEARS	M		2	GROUP 2	USA	2010-12-01
9	T01	DM	T01-001-00009	001-00009	2010-12-01	2010-12-27	001	1949-02-05	61.82067	YEARS	M		2	GROUP 2	USA	2010-12-01
10	T01	DM	T01-001-00010	001-00010	2010-12-01	2010-12-22	001	1947-11-15	63.04723	YEARS	F		1	GROUP 1	USA	2010-12-01
11	T01	DM	T01-001-00011	001-00011	2010-12-01	2010-12-22	001	1942-08-15	68.29843	YEARS	M		2	GROUP 2	USA	2010-12-01
12	T01	DM	T01-001-00012	001-00012	2010-12-02	2010-12-28	001	1938-06-19	72.45722	YEARS	F		1	GROUP 1	USA	2010-12-02
13	T01	DM	T01-001-00013	001-00013	2010-12-06	2010-12-27	001	1951-02-18	59.80014	YEARS	F		2	GROUP 2	USA	2010-12-06
14	T01	DM	T01-001-00014	001-00014	2010-12-06	2011-01-05	001	1932-03-30	78.68857	YEARS	M		2	GROUP 2	USA	2010-12-06
15	T01	DM	T01-001-00015	001-00015	2010-12-06	2010-12-27	001	1946-01-28	64.85695	YEARS	F		2	GROUP 2	USA	2010-12-06
16	T01	DM	T01-001-00016	001-00016	2010-12-06	2011-01-05	001	1946-04-04	64.67625	YEARS	F		2	GROUP 2	USA	2010-12-06
17	T01	DM	T01-001-00017	001-00017	2010-12-06	2010-12-27	001	1949-03-02	61.76591	YEARS	F		1	GROUP 1	USA	2010-12-06
18	T01	DM	T01-001-00018	001-00018	2010-12-07	2011-01-04	001	1942-10-27	68.11499	YEARS	M		1	GROUP 1	USA	2010-12-07
19	T01	DM	T01-001-00019	001-00019	2010-12-07	2011-01-04	001	1947-02-20	63.7974	YEARS	F		2	GROUP 2	USA	2010-12-07
20	T01	DM	T01-001-00020	001-00020	2010-12-07	2011-01-06	001	1947-12-17	62.97604	YEARS	F		2	GROUP 2	USA	2010-12-07
21	T01	DM	T01-001-00021	001-00021	2010-12-07	2010-12-29	001	1950-04-24	60.62423	YEARS	F		1	GROUP 1	USA	2010-12-07
22	T01	DM	T01-001-00022	001-00022	2010-12-08	2011-01-05	001	1947-11-30	63.02533	YEARS	M		2	GROUP 2	USA	2010-12-08
23	T01	DM	T01-001-00023	001-00023	2010-12-08	2011-01-05	001	1939-05-10	71.58385	YEARS	F		2	GROUP 2	USA	2010-12-08
24	T01	DM	T01-001-00024	001-00024	2010-12-08	2011-01-05	001	1937-06-27	73.45106	YEARS	F		2	GROUP 2	USA	2010-12-08
25	T01	DM	T01-001-00025	001-00025	2010-12-08	2011-01-05	001	1942-02-02	68.84873	YEARS	M		1	GROUP 1	USA	2010-12-08
26	T01	DM	T01-001-00026	001-00026	2010-12-08	2011-01-05	001	1936-11-28	74.02875	YEARS	M		2	GROUP 2	USA	2010-12-08

Using ATTRIB before SET sentence can guarantee the sequence of variables in SAS dataset is exactly the same as in the XLS files. KEEP sentence can delete all useless variables in the 'DSN' dataset and keep only variables in DDT in dataset 'out.dm '.

SECTION 2 FURTHER DEVELOPMENTS

Actually, the procedure can be extended to be more automatic. We can add a column named 'Code' in the DDT for the SAS code of the rules indicating how those variables will be derived. An example is given in the below table. In stead of writing the code in SAS, as we did in Section 1.3, we implement the code into the excel file in the 'Code' column.

	A	B	C	D	E	F	G
1	**Domain Dataset name**	**Variable Name**	**Variable Label**	**Type**	**Length**	**Notes**	**Code**
2	DM	STUDYID	Study Identifier	Char	8	DEMOG.study	STUDYID=study;
3	DM	DOMAIN	Domain Abbreviation	Char	2	"DM"	DOMAIN='DM';
4	DM	USUBJID	Unique Subject Identifier	Char	15	Combine study and sub_id variables from DEMOG	USUBJID=trim(left(study))||'-'||trim(left(sub_id));
5	DM	SUBJID	Subject Identifier for the Study	Char	9	DEMOG.sub_id	SUBJID=sub_id;
6	DM	RFSTDTC	Subject Reference Start Date/Time	Char	10	VACCINE.vac_dx where visit_id in ('V01') and vac_yna='Y' and format to ISO8601	if dov_dx ne '' then RFSTDTC=put(input(dov_dx,date9.),yymmdd10.);
7	DM	RFENDTC	Subject Reference End Date/Time	Char	10	TERM.trm_dx. Format to ISO8601.	if trm_dx ne '' then RFENDTC=put(input(trm_dx,date9.),yymmdd10.);
8	DM	SITEID	Study Site Identifier	Char	3	00'||DEMOG.cen_id	SITEID='00'||trim(left(cen_id));
9	DM	BRTHDTC	Date/Time of Birth	Char	10	DEMOG.dob_dx and format to ISO8601	if dob_dx ne '' then BRTHDTC=put(input(dob_dx,date9.),yymmdd10.);
10	DM	AGE	Age in AGEU at Reference Start Date/Time	Num	8	Age in Years: (RFSTDTC - DEMOG.dob_dx + 1)/365.25	if dob_dx ne '' and dov_dx ne '' then AGE=(input(dov_dx,date9.)-input(dob_dx,date9.)+1)/365.25;
11	DM	AGEU	Age Units	Char	6	YEARS	if &name9 ne . then AGEU='YEARS ';
12	DM	SEX	Sex	Char	1	DEMOG.sex_cl M for Male, F for Female and U for Unknown.	SEX=sex_cl;
	DM	RACE	Race	Char	9	use the information in MTA29.DEMOG.race_cl "A" = "Asian" "B" = "Black" "C" = "Caucasian" "H" = "Hispanic" "I" = "American Indian or Alaska Native" "O" = "Other"	RACE=race_cl;

In the same way, we create another macro variable, &code, to store those SAS code and output them with the following code.

```
%macro out;
 data out.dm;
  attrib%do i=1 %to &num;&&name&i   format=&&format&i... label="&&label&i" %end;;
  set dsn;
  %do i=1 to &num; &&code&i &end;
  keep &name;
 run;
%mend out;
%out;
```

So if you combine this part into the program in the previous section, once you finished editing your DDT, you don't need to type any SAS code to derive the variable values.

For those domains which the rules to derive variables are always the same, it's a great benefit to put derivation SAS code in DDT and then call those information into macro variables as well. For instance, most of domains in SDTM datasets, which strictly follow the CDISC standard, are almost the same in various studies.

Another advantage of this method is that it is very convenient for people who are not SAS experts but need to work with SAS, for example, Data Managers. One of the most important responsibilities of Data Managers is to launch logic check for all the trials. For different trials, the check points are always very similar. So it's better to write SAS code into an excel sheet with each cell corresponds to one check point. The only action for Data Manager to take is to add or delete some check points into excel sheet depending on the requirements of trials. Most of the check points can be achieved by some basic SAS sentences such as IF ELSE. After filling the excel sheet, data query could be delivered automatically and send it to investigator by utilizing this existing SAS program.

However, for Analysis Dataset Model (ADaM) datasets, which the definition for variables often need to be changed, it is not an ideal way to implement the SAS code into excel file. Because it is complicate to debug errors. You have to copy your new program into SAS to correct errors first, then modify it in the excel sheet accordingly.

CONCLUSIONS

The advantages of this method are very obvious. First of all, it's faster and easier. No need to type hundreds of variable names, labels and formats from DDT into SAS program; reduces programming time and typing errors. Secondly, the process is highly robust thus the program is highly applicable. Our program can be applied to all domains with minimum modification. What is more, accuracy of the result can be guaranteed. Using this method, not only all the information for the dataset could be updated closely with the DDT, but also the sequence of the variables in SAS datasets would be kept exactly the same as DDT. Mistakes could be easily avoided. Last but not least, this program is user friendly. The program in this method is straightforward, easy to read and be understood; another programmer can modify the program in a very short time.

Of course, this method has its limitation. For those datasets which a standard had been fully developed, this method will work well. However, if the variable name, format, label and value definition of the

datasets need to be changed all the time, this method is not an ideal way since it is complicate to debug.

ACKNOWLEDGMENTS

CONTACT INFORMATION

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

Jianfeng Wang, MSc,
Sanofi Pasteur, Biometry China
Tel: 86-15811002118
jianfeng.wang@ sanofipasteur.com

Linghua Zhou, MSc,
Sanofi Pasteur, Biometry China
Tel: 86-1065685588
linghua.zhou@ sanofipasteur.com