

AUTOSDTM: A Macro to Automatically Map CDASH Data to SDTM

Hao Xu and Hong Chen, McDougall Scientific Ltd.

ABSTRACT

In the pharmaceutical industry, CDISC SDTM is required by various regulatory agencies as the standard data structure for regulatory submission of clinical data. Manually mapping raw data to SDTM domains can be time-consuming and error-prone considering the complexity of clinical data and the strict requirements of the SDTM. However, this process can be much more efficient if the raw data is collected using CDASH standard, allowing for the automatic conversion to the SDTM data structure.

This paper introduces a macro called %AUTOSDTM that can automatically create a SAS® program for each SDTM domain (e.g. dm.sas for DM), that maps CDASH data to SDTM data.

INTRODUCTION

CDISC SDTM consists of specific standard requirements that must be followed in the preparation of the datasets. It is generally not feasible to design a clinical database that will output SDTM datasets; additional programming is required to convert data captured in the clinical study database to SDTM. However, manually mapping raw data to SDTM domains can be time-consuming and error-prone. Fortunately, as CDISC CDASH standards are more and more widely adopted in the database design in clinical trials, SAS macros can be developed for the automatic conversion to the SDTM data structure. This paper introduces a SAS macros %AUTOSDTM that the authors developed to facilitate the CDASH – SDTM conversion.

%AUTOSDTM compares the attributes of CDASH raw datasets with SDTM domains to generate SAS programs that perform the mapping. Each SAS program will contain the code to:

- convert numeric to character variables
- convert date and time to ISO 8601 standard
- create domain-specific variables (e.g. –DY, --BLFL)
- transpose the data sets from latitudinal to longitudinal for Findings domains
- assign variable labels and assign their proper order
- finalize the data sets and write to SDTM library
- create notes to mention unmapped CDASH or SDTM variables

%AUTOSDTM, which sets up the basic frame of SDTM mapping, can minimize the manual work for SAS programmers or, in some cases completely handle some simple domains without any further modifications. This will greatly increase the efficiency and speed of the SDTM conversion process. The following sections will present the technical details about the development of this macro.

GENERAL SETUP

%AUTOSDTM reads in the CDASH standardized raw datasets and gets their attributes (e.g. variables names, formats etc.). These information are used to create conversion (referred as setup in the following sections) programs for SDTM domains.

The macro has one parameter – project, which is the name of the project folder that contains different subfolders.

```
%macro AUTOSDTM(project);
```

```

libname cdash "..\..\&project\cdash";
libname sdtm "..\..\&project\sdtm";
libname master "..\..\&project\master";
libname setup "..\..\&project\setup_sdtm";
...
...
%mend AUTOSDTM;

```

The CDASH and SDTM libraries are used to store CDASH standardized raw datasets and SDTM domains respectively. A master file which maps CDASH to SDTM is created in the master folder (

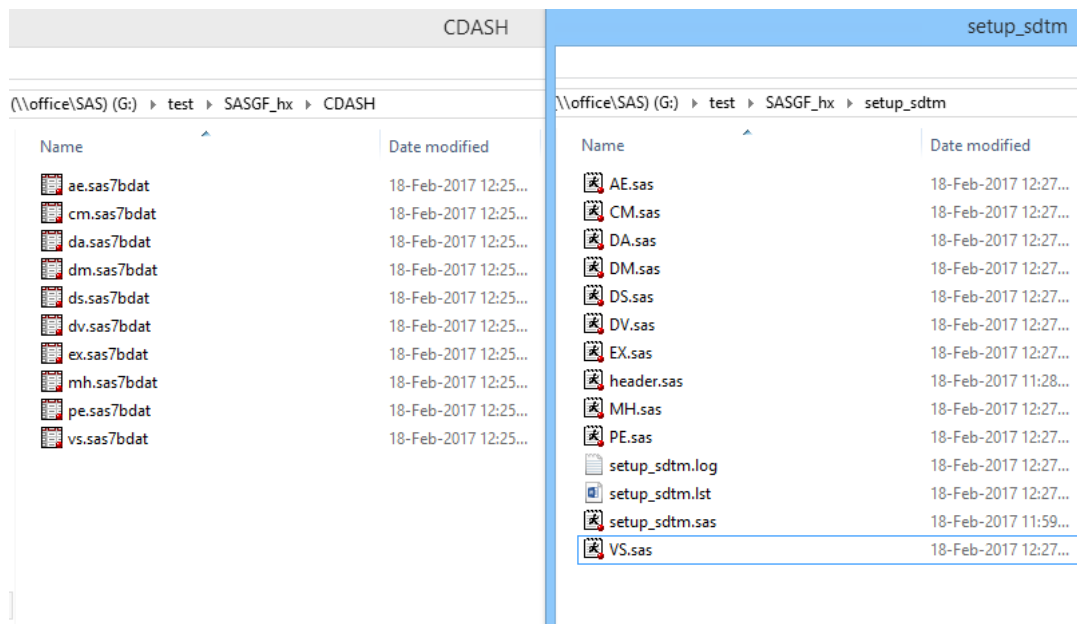
	DOMAIN	CDASHVAR	SDTMVAR	CDASHTEXT	sdtmtype
1	AE	AEACN	AEACN	What action was taken with stu	2
2	AE	AEACNOTH	AEACNOTH	What other action was taken in	2
3	AE	AEBDSYCD	AEBDSYCD	Not Specified	1
4	AE	AEBODSYS	AEBODSYS	Not Specified	2
5	AE	AEDECOD	AEDECOD	Not Specified	2
6	AE	AEDIS		Did the adverse event cause th	
7	AE	AEENDAT	AEENDTC	What date did the adverse even	2
8	AE	AEENTIM	AEENDTC	At what time did the adverse e	2
9	AE	AEHLGT	AEHLGT	Not Specified	2
10	AE	AEHLGTC	AEHLGTC	Not Specified	1
11	AE	AEHLT	AEHLT	Not Specified	2
12	AE	AEHLTCD	AEHLTCD	Not Specified	1
13	AE	AELLT	AELLT	Not Specified	2

Display 1 The master file about mapping CDASH to SDTM

). The master file consists of the domain names, standard CDASH variable names, the corresponding SDTM variable names, and the SDTM variables type. The SETUP_SDTM is the folder where the setup programs are generated. **Error! Reference source not found.** shows the expected output of %AUTOSDTM. For each raw dataset that matches a SDTM domain, %AUTOSDTM will create a setup program, which will be used to transform the raw dataset to SDTM dataset.

	DOMAIN	CDASHVAR	SDTMVAR	CDASHTEXT	sdtmtype
1	AE	AEACN	AEACN	What action was taken with stu	2
2	AE	AEACNOTH	AEACNOTH	What other action was taken in	2
3	AE	AEBDSYCD	AEBDSYCD	Not Specified	1
4	AE	AEBODSYS	AEBODSYS	Not Specified	2
5	AE	AEDECOD	AEDECOD	Not Specified	2
6	AE	AEDIS		Did the adverse event cause th	
7	AE	AEENDAT	AEENDTC	What date did the adverse even	2
8	AE	AEENTIM	AEENDTC	At what time did the adverse e	2
9	AE	AEHLGT	AEHLGT	Not Specified	2
10	AE	AEHLGTC	AEHLGTC	Not Specified	1
11	AE	AEHLT	AEHLT	Not Specified	2
12	AE	AEHLTCD	AEHLTCD	Not Specified	1
13	AE	AELLT	AELLT	Not Specified	2

Display 1 The master file about mapping CDASH to SDTM



Display 2 The setup programs generated by %AUTOSDTM

STEP 1: READ IN THE CDASH RAW DATASETS AND PREPARE FOR GENERATING SETUP PROGRAMS

READ IN CDASH RAW DATASETS

First, %AUTOSDTM reads in the CDASH Raw Datasets and gets the attributes. These attributes information are merged with the master file to obtain the corresponding SDTM variables.

Table 1 CDASH Datasets Attributes and Corresponding SDTM Variables **Error! Reference source not found.** shows the attributes we get in this stage, using AE domain as an example. These information will be used later to generate the setup programs.

DOMAIN	CDASH VARIABLE	CDASH VAR TYPE	CDASH FORMAT	SDTM VARIABLE	SDTM VAR TYPE
AE	AEACN	1	ACN.	AEACN	2
AE	AEENDAT	1	YYMMDD	AEENDTC	2
AE	AEENTIM	1	TIME	AEENDTC	2
AE	AEONGO	2	\$	AEENRF or AEENRTPT and AEENTPT	.
AE	AEREL	2	\$	AEREL	2
AE	AESPID	1	BEST	AESPID	2
AE	AESTDAT	1	DATE	AESTDTC	2
AE	AESTTIM	1	TIME	AESTDTC	2
AE	AETERM	2	\$	AETERM	2
AE	AEYN	2	\$.

Table 1 CDASH Datasets Attributes and Corresponding SDTM Variables

GET THE DOMAIN INFORMATION FOR THIS PROJECT

%AUTOSDTM prepares a dataset containing all the domain names applicable for this project (Table 2 Domain information for this project **Error! Reference source not found.**), based on the CDASH datasets. These domain names are assigned to a series of macro variables, which are used later to create setup programs.

OBS	DOMAIN	CLASS
1	AE	Events
2	CM	Interventions
3	DA	Findings
4	DM	Special-Purpose
5	DS	Events
6	DV	Events
7	EX	Interventions
8	MH	Events
9	PE	Findings
10	VS	Findings

Table 2 Domain information for this project

The DOMAIN names and relative CLASS information are then assigned to macro variables (&&domain&i and &&class&i):

```
data _null_;
  set projdomain end=eof;
  call symput ('domain'||strip(put(_n_, best.)),compress(domain));
  call symput ('class'||strip(put(_n_, best.)), compress(class));
  if eof then call symput('ndomain', strip(put(_n_, best.)));
run;
```

A macro variable &ndomain, denotes the number of the domains, is also created.

STEP 2: GENERATE THE SETUP PROGRAMS

1. DETERMINE IF SETUP PROGRAMS ALREADY EXIST

Since the macro writes setup programs in SETUP_SDTM folder, there exists the potential risk that you may overwrite an existing SDTM setup program unintentionally. Before generating the setup program for each domain, %AUTOSDTM check whether the setup program already exists. If yes, it will not be executed for this domain. Otherwise, it gets the attributes information for each domain as shown in Table 1 and proceeds to generating the setup program.

```
%do i=1 %to &ndomain;
  %if %sysfunc(fileexist(.\&&domain&i...sas))=0 %then %do;
```

.....

```

%end;
%end; *** End Do Loop;

```

2. TRANSFORM GENERAL VARIABLES BASED ON THE VARIABLE TYPE.

The following program shows how the setup programs are generated based on the CDASH and SDTM variable attributes, as shown in Figure 1 The generated setup program for AE domain **Error!**
Reference source not found.

```

data &&domain&i.;
  set &&domain&i. end=last;
  by domain cdashvar;
  length mapfl 8;
  file "&&domain&i...sas";
  if _n_ =1 then do; *** Writing the header for setup program;
    put *****;"/
      *** Author: ;"/
      *** Date: %sysfunc(date(), yymmdd10.) ;"/
      *** Input: ;"/
      *** Output: SDTM.&&domain&i.. ;"/
      *****;"/
      "%include header;"/
      "option validvarname=v7;" //;

    *** eg. use AE1 as dataset name;
    put "data " domain +(-1) "1;"/
      " set cdash." domain ";" /
      " %study(&&domain&i);";

  end;

*** if CDASH and SDTM names are the same, transform the variable based on the
variable type;
if cdashvar=sdtmvar then do;
  put +2 "%transvar(invar=" cdashvar ",intype=" type ",outvar=" sdtmvar
    ",outtype=" sdtmtype");";
  mapfl = 1;
end;
*** deal with DTC variable - date part;
else if substr(cdashvar,length(cdashvar)-2)='DAT' and
  substr(sdtmvar,length(sdtmvar)-2)='DTC' then do;
  put +2 "%sdtmdt(indate=" cdashvar ", intype=" type ", outdate=" sdtmvar ");";
  mapfl = 1;
end;

*** deal with DTC variable - time part;
else if substr(cdashvar,length(cdashvar)-2)='TIM' and
  substr(sdtmvar,length(sdtmvar)-2)='DTC' then do;
  put +2 "%sdtmtm(intime=" cdashvar ", intype=" type ", outdate=" sdtmvar ");";
  mapfl = 1;
end;

```

```

1 *****;
2 ** Author: ;
3 ** Date: 2017-02-18 ;
4 ** Input: ;
5 ** Output: SDTM.AE ;
6 *****;
7
8 %include header;
9
10 option validvarname=v7;
11
12
13 data AE1;
14 set cdash.AE ;
15 %study(AE);
16 %transvar(invar=AEACN , intype=1 , outvar=AEACN , outtype=2 );
17 %transvar(invar=AEACNOTH , intype=2 , outvar=AEACNOTH , outtype=2 );
18 %sdtmdt(indate=AEENDAT , intype=1 , outdate=AEENDTC );
19 %sdtmtm(intime=AEENTIM , intype=1 , outdate=AEENDTC );
20 %transvar(invar=AEREL , intype=2 , outvar=AEREL , outtype=2 );
21 %transvar(invar=AESCONG , intype=2 , outvar=AESCONG , outtype=2 );
22 %transvar(invar=AESDISAB , intype=2 , outvar=AESDISAB , outtype=2 );
23 %transvar(invar=AESDTH , intype=2 , outvar=AESDTH , outtype=2 );
24 %transvar(invar=AESEV , intype=2 , outvar=AESEV , outtype=2 );
25 %transvar(invar=AESHOSP , intype=2 , outvar=AESHOSP , outtype=2 );
26 %transvar(invar=AESLIFE , intype=2 , outvar=AESLIFE , outtype=2 );
27 %transvar(invar=AESMIE , intype=2 , outvar=AESMIE , outtype=2 );
28 %transvar(invar=AESPID , intype=1 , outvar=AESPID , outtype=2 );
29 %sdtmdt(indate=AESDAT , intype=1 , outdate=AESDTC );
30 %sdtmtm(intime=AESTTIM , intype=1 , outdate=AESTDTC );
31 %transvar(invar=AETERM , intype=2 , outvar=AETERM , outtype=2 );
32 run;
33

```

Figure 1 The generated setup program for AE domain

From this example, you can see that a series of small macros are employed to perform the transformation:

1. %transvar is employed to transform general variables except date/time. %transvar uses different approaches to do the transformation based on different CDASH variable type.
2. %sdtmda is used to transform --DAT to --DTC variable.
3. %sdtmtm is used to combine --TIM and --DTC variable.

Please note these small macros are not part of %AUTOSDTM. They are prepared in the header.sas. By using these macros, the setup program is much easier to read and maintain. Another advantage is that these small macros can be modified specifically to deal with different projects. Following is the code for the macro %transvar. A temporary variable &invar._ is created to deal with the situation when &invar and &outvar share the same variable name.

```

%macro transvar(invar,intype,outvar,outtype);
  %if &outtype = 2 %then %do;
    length &invar._ $200;
    %if &intype = 1 %then %do;
      &invar._=upcase(putn(&invar,vformat(&invar)));
    %end;
    %else %if &intype = 2 %then %do;

```

```

        &invar._=strip(uppercase(&invar));
    %end;
%end;
%else %if &outtype = 1 %then %do;
    length &invar._ 8;
    %if &intype = 1 %then %do;
        &invar._=&invar;
    %end;
    %else %if &intype = 2 %then %do;
        &invar._=input(&invar,best.);
    %end;
%end;
drop &invar;
rename &invar._=&outvar;
%mend transvar;

```

3. DEALING WITH SPECIFIC VARIABLES FOR EACH DOMAIN

Since SDTM requires derived variables that are not directly captured in CDASH, additional steps must be taken to create these variables. Specific variables that are not in our mapping master file, are taken care of for each specific domain. For example, the following code creates DTHFL variable for the DM domain.

```

**** DM Domain;
if domain = 'DM' then do;
    ** for DTHFL;
    if last then do;
        put +2 "if DTHDTC>' ' then DTHFL = 'Y!;"/
            +2 "else DTHFL = 'N!;";
    end;
end;

```

4. TRANSFORM DATA FROM HORIZONTAL TO VERTICAL (VS AND EG)

Finding domains like VS and EG normally have fixed tests, for example, blood pressure for vital signs. It is more convenient to design the database in a horizontal way (Display 2 The setup programs generated by %AUTOSDTM). In order to transform these variables to vertical using different --TESTCD, you can standardize these variable names using the --TESTCD terminology, and map these variables to --ORRES in the master file (Display 3 The horizontal data for Vital Signs).

	pulse	pulseu	resp	respu	sysbp	sysbpu	diabp	diabpu	weight	weightu
1	70	1	20	1	100	1	60	1	70	1
2	71	1	19	1	101	1	63	1	70.4	1
3	76	1	20	1	98	1	66	1	.	.
4	75	1	20	1	103	1	59	1	.	.
5	80	1	18	1	100	1	65	1	65	1
6	78	1	18	1	91	1	62	1	65	1
7	77	1	19	1	98	1	58	1	.	.

Display 3 The horizontal data for Vital Signs

SAS Universal Viewer - [g:\test\sasgf_hx\master\cdash_sdtm.sas7bdat]

File Tools Window Help

Address

Library CDASH_SDTM

Freeze Hide Show... Format Filter... Font... Find

Table View

	DOMAIN	CDASHVAR	SDTMVAR	CDASHTEXT	sdtmtype
234	VS	PULSE	VSORRES		2
235	VS	PULSEU	VSORRESU		2
236	VS	RESP	VSORRES		2
237	VS	RESPU	VSORRESU		2
238	VS	SYSBP	VSORRES		2
239	VS	SYSBPU	VSORRESU		2
240	VS	TEMP	VSORRES		2
241	VS	TEMPU	VSORRESU		2

Display 4 The master file to map VS data from horizontal to vertical

The generated setup program looks like this:

```

VS.sas
13 data VS1;
14   set cdash.VS ;
15   %study(VS);
16   %transvar(invar=VISIT , intype=2 , outvar=VISIT , outtype=2 );
17   %transvar(invar=VISITNUM , intype=1 , outvar=VISITNUM , outtype=1 );
18   %sdtmdt(indate=VSDAT , intype=1 , outdate=VSDTC );
19   VSTESTCD = put(VSTEST,VSTEST.);
20 run;
21
22
23 data VS1;
24   set VS1;
25   %transvar(invar=WEIGHTU , intype=1 , outvar=VSORRESU , outtype=2 );
26   %transvar(invar=WEIGHT , intype=1 , outvar=VSORRES , outtype=2 );
27   VSSTRESC = VSORRES; ** NEED Double Check;
28   VSSTRESN = input(VSORRES, best.);
29   VSSTRESU = VSORRESU;
30   output;
31   %transvar(invar=TEMPU , intype=1 , outvar=VSORRESU , outtype=2 );
32   %transvar(invar=TEMP , intype=1 , outvar=VSORRES , outtype=2 );
33   VSSTRESC = VSORRES; ** NEED Double Check;
34   VSSTRESN = input(VSORRES, best.);
35   VSSTRESU = VSORRESU;
36   output;
37   %transvar(invar=SYSBPU , intype=1 , outvar=VSORRESU , outtype=2 );
38   %transvar(invar=SYSBP , intype=1 , outvar=VSORRES , outtype=2 );
39   VSSTRESC = VSORRES; ** NEED Double Check;
40   VSSTRESN = input(VSORRES, best.);
41   VSSTRESU = VSORRESU;
42   output;
43   %transvar(invar=RESPU , intype=1 , outvar=VSORRESU , outtype=2 );
44   %transvar(invar=RESP , intype=1 , outvar=VSORRES , outtype=2 );
45   VSSTRESC = VSORRES; ** NEED Double Check;
46   VSSTRESN = input(VSORRES, best.);
47   VSSTRESU = VSORRESU;
48   output;

```

Figure 2 Setup program to transform VS from horizontal to vertical

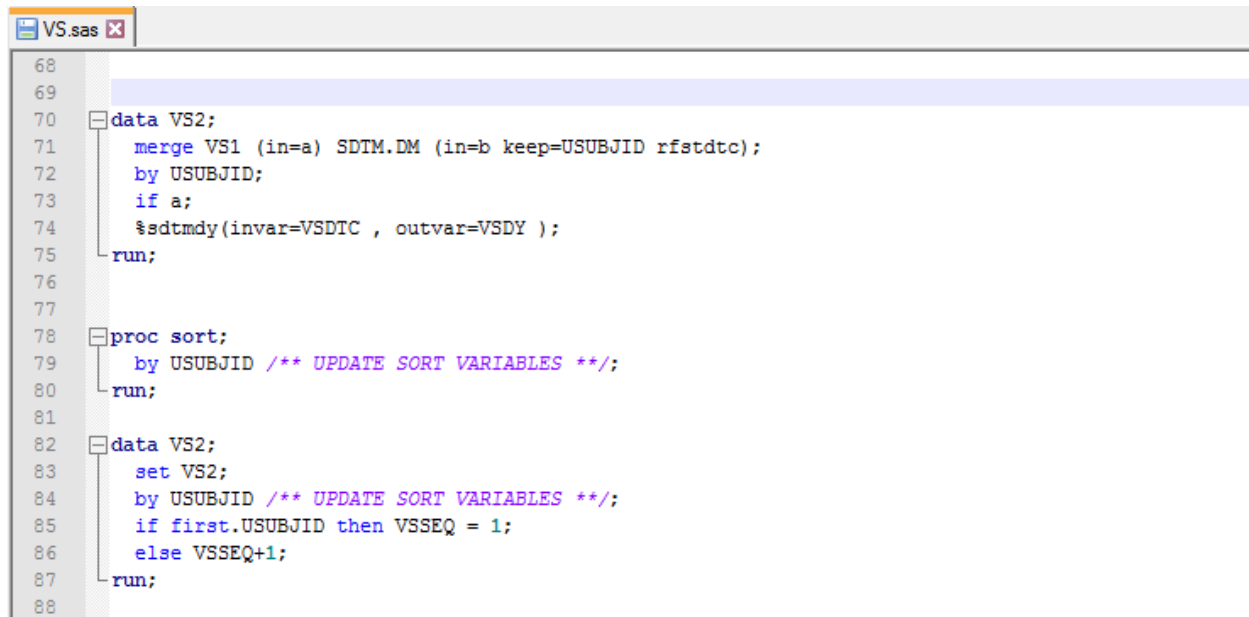
5. CALCULATE --DY AND --SEQ VARIABLES

Study Day (--DY) is another example of derived variable that is not captured in CDASH. %AUTOSDTM can prepare program to calculate the corresponding --DY variables for all -DTC variables.

```
proc sort data=&&domain&i. out=&&domain&i.._dtc nodupkey;
  by sdtmvar;
  where substr(sdtmvar,length(sdtmvar)-2)='DTC';
run;
data &&domain&i.._dtc;
  set &&domain&i.._dtc;
  dyvar = substr(sdtmvar,1,length(sdtmvar)-3)||'DY';
run;
```

--SEQ variables are required variables in almost every domain except for DM, SV and trial design domains. %AUTOSDTM generates the programs to calculate the --SEQ variable for these domains:

The generated programs is presented below:



```
VS.sas
68
69
70 data VS2;
71   merge VS1 (in=a) SDTM.DM (in=b keep=USUBJID rfstdtc);
72   by USUBJID;
73   if a;
74   %sdtmdy(invar=VSDTC , outvar=VSDY );
75 run;
76
77
78 proc sort;
79   by USUBJID /** UPDATE SORT VARIABLES **/;
80 run;
81
82 data VS2;
83   set VS2;
84   by USUBJID /** UPDATE SORT VARIABLES **/;
85   if first.USUBJID then VSSEQ = 1;
86   else VSSEQ+1;
87 run;
88
```

Figure 3 Setup program to calculate --DY and --SEQ

6. CALCULATE --BLFL VARIABLES FOR SOME FINDINGS DOMAINS

For some of the Findings domains (LB, VS, and FA), --BLFL is an Expected variable. It is normally defined as the last non-missing record before the treatment. %AUTOSDTM can generate the program to calculate the --BLFL for these domains.

```

VS.sas
89
90 proc sort data=VS2 out=base(keep=usubjid VStestcd VSseq /** UPDATE SORT VARIABLES **/ );
91   by usubjid VStestcd /** UPDATE SORT VARIABLES **/;
92   where /** UPDATE WHERE CLAUSE **/;
93 run;
94
95 data base;
96   set base;
97   by usubjid VStestcd /** UPDATE SORT VARIABLES **/;
98   if last.VStestcd;
99   keep usubjid VSseq;
100 run;
101 proc sort;
102   by usubjid VSseq;
103 run;
104
105 proc sort data=VS2;
106   by usubjid VSseq;
107 run;
108
109 data VS2;
110   merge VS2 base(in=b);
111   by usubjid VSseq;
112   if b then VSb1f1 = 'Y';
113 run;
114
115 data VS;
116   set VS2;
117   keep /* NEED MANUALLY INPUT*/;
118 run;
119
120 %sdtmfinal(domain=VS,class=);
121
122
123 proc sort data=VS out=sdtm.VS nodupkey;
124   by USUBJID VSSEQ;
125 run;
126

```

Figure 4 Setup program to calculate –BLFL variable

7. FINALIZE THE DATA AND SAVE TO SDTM LIBRARY

The last step is to finalize the data set before save the data into SDTM library. The following things are accomplished:

1. Keep variables only for SDTM domain
2. Assign the labels from the SDTM metadata master file (SDTM V1.4)
3. Cross check the dataset with SDTM master file to confirm
 - a. if the variable type matches with SDTM master file
 - b. if the variable name is correct
 - c. if any required or expected variables in this SDTM domain are not included
 - d. if any variables have unexpected control terminology
4. Assign the order of the variables
5. Adjust the length of the characteristic variables, remove leading and trailing spaces of the characteristic variables
6. Assign domain labels
7. Write finalized data to SDTM library

```

data _null_;
  file "&&domain&i...sas" mod;

```

*** Notify the programmer to manually update the variables that need to be kept;

```

put "data &&domain&i.;"/
  " set &&domain&i..2;"/
  " keep /* NEED MANUALLY INPUT*/;"/
"run;"/;

*** Step 2-6 is accomplished by %stdmfinal macro;
put "%sdtmfinal(domain=&&domain&i.,class=)";"/;

*** Write to SDTM library;
put "proc sort data=&&domain&i. out=sdtm.&&domain&i. nodupkey;"/
  " by USUBJID &&domain&i..SEQ;"/
  "run;"/;

run;

```

8. NOTIFY UNSOLVED CDASH VARIABLES

Not all clinical trials are designed in the same ways. Even for a database that is designed per the CDASH standards, there is no guarantee that all data will map nicely to SDTM. Although %AUTOSDTM can handle most of the situations, it is common to have special variables that need the programmer to manually take care of. A comment is generated by the following program to notify the programmer on variables are still untransformed and require further attention.

```

data &&domain&i.._cdash;
  set &&domain&i;
  if mapfl~=1;
  if cdashvar in ('SUBJID') then delete;
  keep cdashvar;
run;

%local listcdash;
proc sql noprint;
  select cdashvar into :listcdash
  separated by ' '
  from &&domain&i.._cdash;
quit;

data _null_;
  file "&&domain&i...sas" mod;
  put "*****";
  %if &listcdash~= %then "**** Following variables in CDASH have not be
transformed to SDTM yet: &listcdash ;"/;
  "*****";
run;

```

CONCLUSION

The macro %AUTOSDTM can be used to automatically create setup programs for SDTM domains based on the CDASH standardized raw datasets. By applying this approach, the efficiency of transforming CDASH raw datasets to SDTM domains can be greatly improved. The key benefits from using this macro are:

1. Automatically generate the setup programs for SDTM domains with minimum modifications required.
2. CRF Annotation and data specification document for SDTM are more straightforward, considering the similarity of CDASH and SDTM standards.

There are some limitations/restrictions of %AUTOSDTM:

1. Setup programs for trial design domains cannot be automatically generated.

2. The raw datasets need to be CDASH compliant, at least 'CDASH-like'.
3. SDTM terminology need to be applied for the raw data variable values.

REFERENCES

CDISC CDASH Team, 2011. Clinical Data Acquisition Standards Harmonization (CDASH) User Guide, V 1-1.1.

CDISC Submission Data Standards Team, 2013. Study Data Tabulation Model Implementation Guide: Human Clinical Trials Version 3.2.

Chen H., 2014. Converting Clinical Database to SDTM: The SAS® Implementation. SGF 2014.

ACKNOWLEDGMENTS

We would like to thank the statisticians in McDougall Scientific Ltd. for their support and comments.

APPENDIX

```

/*****
/***** AUTOSDTM: A Macro to Map CDASH Data to SDTM *****/
/*****

%macro AUTOSDTM(project);

libname cdash "..\..\&project\cdash";
libname sdtm "..\..\&project\sdtm";
libname master "..\..\&project\master";
libname setup "..\..\&project\setup_sdtm";

option validvarname = v7;

*-----;
***- 1. Read in the CDASH Datasets and prepare for generating set up programs;
*-----;

*-----;
**** 1.1 read in CDASH datasets;

**** Use PROC CONTENTS to get Attribute of CDASH datasets;
proc contents noprint data=cdash._all_ out=cdashatr;
run;

data cdashatr;
  set cdashatr;
  keep memname cdashvar type format;
  rename memname = domain;
  cdashvar=upcase(name);
run;
proc sort;
  by domain cdashvar;
run;

```

```

**** master file of mapping CDASH to SDTM;
proc sort data= master.cdash_sdtm out=cdash_sdtm(keep=domain cdashvar sdtmvar
sdtmtype);
  by domain cdashvar;
run;

proc sort data= master.sdtmv14 out=sdtmv14;
  by domain varname;
run;

data cdashatr;
  merge cdashatr(in = a) cdash_sdtm;
  by domain cdashvar;
  if a;
run;

*-----;
**** 1.2 get the domains for this project and start preparing for setup program;
proc sort data=cdashatr out=projdomain(keep=domain) nodupkey;
  by domain;
run;

proc sort data=sdtmv14 out=domainclass(keep=domain class) nodupkey;
  by domain;
  where domain>';
run;

data projdomain;
  merge projdomain(in=a) domainclass;
  by domain;
  if a;
run;

data _null_;
  set projdomain end=eof;
  call symput ('domain' || strip(put(_n_, best.)), compress(domain));
  call symput ('class' || strip(put(_n_, best.)), compress(class));
  if eof then call symput('ndomain', strip(put(_n_, best.)));
run;

*-----;
***----- 2. Write the setup program for each domain -----;
*-----;

%do i=1 %to &ndomain;

*-----;
**** 2.1 Check if the setup programs exist or not, if yes, stop execute -----;

%if %sysfunc(fileexist(.\&&domain&i...sas))=0 %then %do;

*-----;
**** Get the attribute information for each domain;
data &&domain&i..;
  set cdashatr;
  if domain = "&&domain&i..";

```

```

if index(strip(sdtmvar),' ') then sdtmvar = ''; ** only keep SDTMVARS which can be
1 to 1 map;
run;

*-----;
* 2.2 Generate setup program: transform the variables based on CDASH variable type;

data &&domain&i.;
set &&domain&i. end=last;
by domain cdashvar;
length mapfl 8; ** MAPFL is used to track variables that have not been mapped;
file "&&domain&i...sas";
if _n_ =1 then do; *** Writing the header for setup program;
  put "*****;"/
     "** Author:                               ;"/
     "** Date:      %sysfunc(date(), yymmdd10.)  ;"/
     "** Input:                               ;"/
     "** Output:      SDTM.&&domain&i..         ;"/
     "*****;"/
     "%include header;"/;

  *** eg. use AE1 as dset name;
  put "data " domain +(-1) "1;"/
     " set cdash." domain ";"/
     " %study(&&domain&i);";
end;

*** if CDASH and SDTM name are the same, transform the variable based on the
variable type;
if cdashvar=sdtmvar then do;
  put +2 "%transvar(invar=" cdashvar ", intype=" type ", outvar=" sdtmvar ",
outtype=" sdtmtype");";
  mapfl = 1;
end;
*** deal with DTC variable - date part;
else if substr(cdashvar,length(cdashvar)-2)='DAT' and
substr(sdtmvar,length(sdtmvar)-2)='DTC' then do;
  put +2 "%sdtmdt(indate=" cdashvar ", intype=" type ", outdate=" sdtmvar ");";
  mapfl = 1;
end;

*** deal with DTC variable - time part;
else if substr(cdashvar,length(cdashvar)-2)='TIM' and
substr(sdtmvar,length(sdtmvar)-2)='DTC' then do;
  put +2 "%sdtmtm(intime=" cdashvar ", intype=" type ", outdate=" sdtmvar ");";
  mapfl = 1;
end;

*** The following are dealing with specific variables for each domain - this part
will be modified based on practical situations;
**** CM Domain;
if domain = 'CM' then do;
  if cdashvar = 'CMDSTXT' then do; *** CMDSTXT mapped to CMDDOSE or CMDOSTXT;
    if type = 2 then put +2 "CMDOSTXT = CMDSTXT;";
    else if type = 1 then put +2 "CMDDOSE = CMDSTXT;";
    mapfl = 1;
  end;
end;

**** DM Domain;
if domain = 'DM' then do;
  ** for DTHFL;

```

```

    if last then do;
    put +2 "if DTHDTC>' then DTHFL = 'Y';"/
      +2 "else DTHFL = 'N';";
    end;
end;

*** for Findings;
if "&&class&i."='Findings' then do;
  if last then do;
    put +2 "&&domain&i..TESTCD = put(&&domain&i..TEST,&&domain&i..TEST.);";
  end;
end;

if last then do;
  put "run;"/;
end;
run;

*-----;
*** 2.3 transform data from horizontal to vertical - for VS and EG;

%if &&domain&i. = VS or &&domain&i. = EG %then %do;

proc sort data=&&domain&i;
  by domain descending cdashvar;
run;

data &&domain&i.;
  set &&domain&i. end=last;
  by domain descending cdashvar;
  length tempcdash $200;
  retain tempcdash;
  file "&&domain&i..sas" mod;
  if _n_ =1 then do; *** Writing the header for setup program;
    put "data " domain +(-1) "1;"/
      " set " domain +(-1) "1;" ;
  end;
  if cdashvar~=sdtmvar and substr(sdtmvar,3) in ('ORRES','ORRESU') then do;

    put +2 "%transvar(invar=" cdashvar ", intype=" type ", outvar=" sdtmvar ",
outtype=" sdtmtype");";

    if substr(sdtmvar,3)= 'ORRES' then
      put " &&domain&i..STRESC = ; ** NEED MANUALLY UPDATE;"/
        " &&domain&i..STRESN = ; "/
        " &&domain&i..STRESU = ; "/
        " output;";
    mapfl = 1;
    tempcdash = cdashvar;
  end;

  if last then do;
    put "run;"/;
  end;
run;
%end;

*-----;

```

```

*** 2.4 Merge with DM and calculate --DY, and also --SEQ - except DM/SV and trial
design domain;

%if &&domain&i. ~= DM and &&domain&i. ~= SV %then %do;
proc sort data=&&domain&i. out=&&domain&i.._dtc nodupkey;
  by sdtmvar;
  where substr(sdtmvar,length(sdtmvar)-2)='DTC';
run;
data &&domain&i.._dtc;
  set &&domain&i.._dtc;
  dyvar = substr(sdtmvar,1,length(sdtmvar)-3)||'DY';
run;

data _null_;
  set &&domain&i.._dtc end=last;
  by domain cdashvar;
  file "&&domain&i...sas" mod;
  if _n_ =1 then do;
    *** a new data set to merge with DM domain;
    put "data " domain +(-1) "2;"/
      " merge " domain +(-1) "1 (in=a) SDTM.DM (in=b keep=USUBJID
rfstdtc);" /
      " by USUBJID;"/
      " if a;";
  end;
  put " %sdtmdy(invar=" sdtmvar ", outvar=" dyvar ");"; *** calculate the --DY
variable based on --DTC;
  if last then do;
    put "run;"/;

    *** calculate --SEQ;
    put "proc sort;"/
      " by USUBJID /** UPDATE SORT VARIABLES **/;"/
      "run;"/

      "data " domain +(-1) "2;"/
      " set " domain +(-1) "2;"/
      " by USUBJID /** UPDATE SORT VARIABLES **/;"/
      " if first.USUBJID then &&domain&i..SEQ = 1;"/
      " else &&domain&i..SEQ+1;"/
      "run;"/;
  end;
run;
%end;

*-----;
*** 2.5 Calculate baseline flag for some of findings domains (LB/VS/FA);
*** Note: the sort and where clause need to be manually updated later;

%if &&domain&i=LB or &&domain&i=VS or &&domain&i=FA %then %do;
data _null_;
  file "&&domain&i...sas" mod;
  put
"proc sort data=&&domain&i..2 out=base(keep=usubjid &&domain&i..testcd
&&domain&i..seq /** UPDATE SORT VARIABLES **/ );"/
" by usubjid &&domain&i..testcd /** UPDATE SORT VARIABLES **/;"/
" where /** UPDATE WHERE CLAUSE **/;"/
"run;"/;

  put "data base;"/

```



```

" set base;"/
" by usubjid &&domain&i..testcd /** UPDATE SORT VARIABLES **;/"/
" if last.&&domain&i..testcd;"/
" keep usubjid &&domain&i..seq;"/
"run;"/
"proc sort;"/
" by usubjid &&domain&i..seq;"/
"run;"/
"proc sort data=&&domain&i..2;"/
" by usubjid &&domain&i..seq;"/
"run;"/

put "data &&domain&i..2;"/
" merge &&domain&i..2 base(in=b);"/
" by usubjid &&domain&i..seq;"/
" if b then &&domain&i..blfl = 'Y';"/
"run;"/

run;
%end;

*-----;
*** 2.6 finalized the data and write to SDTM library;
data _null_;
file "&&domain&i...sas" mod;
*** Notify the programmer to manually update the variables that need to be kept;
put "data &&domain&i.;"/
" set &&domain&i..2;"/
" keep /* NEED MANUALLY INPUT*/;"/
"run;"/

*** This macro will finalize the SDTM dataset - assign label, adjust char length
etc..;
put "%sdtmfinal(domain=&&domain&i.,class=)";"/;

*** Write to SDTM library;
put "proc sort data=&&domain&i. out=sdtm.&&domain&i. nodupkey;"/
" by USUBJID &&domain&i..SEQ;"/
"run;"/;

run;

*-----;
*** 2.7 Notify unsolved CDASH variables;
data &&domain&i.._cdash;
set &&domain&i;
if mapfl~=1;
if cdashvar in ('SUBJID') then delete;
keep cdashvar;
run;

%local listcdash;
proc sql noprint;
select cdashvar into :listcdash
separated by ' '
from &&domain&i.._cdash;
quit;

data _null_;
file "&&domain&i...sas" mod;
put "*****";
%if &listcdash~= %then "**** Following variables in CDASH have not be transformed
to SDTM yet: &listcdash ";"/;

```

```
*****";  
run;  
%end;  
%end; ** end do loop;  
  
%mend AUTOSDTM;
```

CONTACT INFORMATION

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

Hao Xu
McDougall Scientific Ltd.
789 Don Mills Road, Suite 802, Toronto, ON
M3C 1T5
hxu@mcdougallscientific.com

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are trademarks of their respective companies.