
An automated code translation approach in SDTM specifications using %translate macro

Aman Bahl, Syneos Health, Ontario, Canada

Hrideep Antony, Syneos Health, North Carolina, USA

March 11th 2020

Author's Bio

Hrideep Antony is a Principal Statistical Programmer with Syneos Health, and has over 14 years of extensive SAS experience. Hrideep enjoys developing and providing innovative/efficacious solutions to streamline the FDA submission process. He has been given opportunities to present at multiple SAS conferences in the past. Hrideep enjoys spirituality, music, travelling, volunteering at nursing homes/rehabilitation facilities, learning and sharing knowledge.

Aman Bahl is an Associate Director, Statistical Programming with Syneos Health, and has over 18 years of Technology and Data services experience including 15 years of Clinical industry experience in the areas of project management, team management, statistical programming and other statistical services. Aman enjoys outdoor sports, trekking, travelling and volunteering.

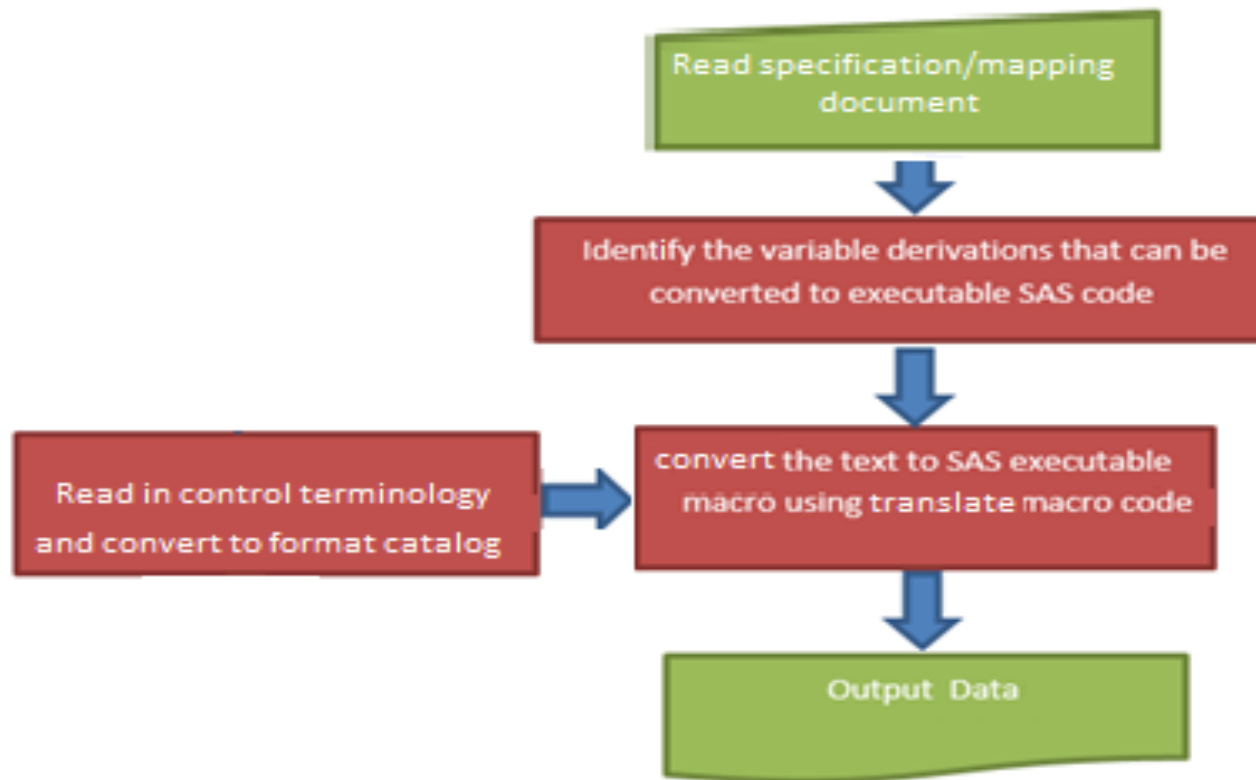
Agenda

- ❖ Introduction
- ❖ Process flow chart
- ❖ Specification language standardization
- ❖ Programming process
- ❖ Conclusion

Introduction

- ❖ Specification could be more than just a specification.
- ❖ Standardized variable derivation approach.
- ❖ About 50 percent of SDTM variable derivations involve directly reading raw data variables or simple subset of raw data.
- ❖ Converting SDTM specification language to a programming script

Process flow chart



Specification language standardization

Standardizing the RFICDTC (Date/Time of Informed Consent) derivation

Index	ALGORITHM
1	Equal to SBJDTM in raw data SUBINFO where INFTTYPE value='ICDT'
2	Set to SBJDTM in raw data SUBINFO when INFTTYPE value='ICDT'
3	Set to SUBINFO. SBJDTM where INFTTYPE ≠ 'ICDT'

- Index 3 is a more standardized approach relative to the other two derivations as the representation is more direct with the usage of the period to separate the data and the variable
- Index 3 can easily be programmatically converted to an executable SAS code.

SUBINFO Raw data			Output dataset	
SUBJID	SBJDTM	INFTTYPE	SUBJID	RFICDTC
6503	2020-01-08	ICDT	6503	2020-01-08
5672	2020-01-07	ICDT	5672	2020-01-07
5877	2020-01-07	ICDT	5877	2020-01-07
5593	2020-01-08	ICDT	5593	2020-01-08
5588	2020-01-07	RAND		
6139	2020-01-07	RAND		
5845	2019-12-17	RAND		
5812	2019-12-06	RAND		
5771	2019-12-05	RAND		

Specification language standardization examples

DATASET	VARIABLE	LABEL	ALGORITHM	Translated SAS code language
DM	DOMAIN	Domain Abbreviation	Set to "DM"	translate to : DOMAN="DM"
DM	RFXSTDTC	Date/Time of First Study Treatment	Set to RFXSTDTC	translate to : RFXSTDTC=RFXSTDTC
SUPPDM	ETHYSC5	Cuban	Set to DMx1009.ETHNICSCAT_35	translate to : data _trans_1; set DMx1009; ETHYSC5=ETHNICSCAT_35;run;
DM	DTHFL	Subject Death Flag	SET to "Y" if DSx001.DSDECOD="DEATH"	translate to : data _trans_4; set DSx001; if DSDECOD="DEATH" ; DTHFL="Y";run;

Example One: Algorithm starts with a simple “Set to”. The algorithm looks for the succeeding text after the “Set to” statement. If the text is within a quote, it gets translated to the variable=”TEXT”

Example Two: Algorithm that starts with “Set to” but has a text without quotes as in the example of RFXSTDTC variable. In this case, the value of the variable named RFSTDTC will be assigned to RFXSTDTC.

Example Three: Algorithm has a “period” and begins with a “Set to”. Translate macro will convert this statement to a data step. The ETHYSC5 variable gets assigned to the variable ETHNICSCAT_35 which is in the DMx1009 dataset

Example Four: Algorithm has a “period” and begins with a “Set to”. Translate macro converts this statement to a data step. The ETHYSC5 variable gets assigned to the variable ETHNICSCAT_35 which is in the DMx1009 dataset.

Programming process

Step 1: Read in the specification documents and identify the derivations that can be programmatically processed.

```
%*=====;
*   Read in the specification document and convert the file to a SAS dataset   ;
%*=====;

libname spec xlsx "&specpath" ;

data spec1;
set spec.&specname;;
%*=====;
*   read in the Derivation algorithm and process the data   ;
%*=====;
STD_alg=trim(&algorithm);
rev_std_alg=compress(reverse(STD_alg));
%*=====;
*   Split the text in the algorithm to multiple split variables for identifying the derivation logic.
    Note that the maximum allowable split words are 20 words in the cuurent version.
    The future versions of the utility will allow more texts to be processed ;
%*=====;
array split (*)$50 split1-split20;
do i = 1 to 20;
    split(i)= scan(STD_alg,i);
end;
```

Programming process cont..

Step 2: A “Type” variable is assigned with a value of “Internal” if the logic does not involve external library references and a Value of “External” if the derivation logic involves external file references. “Set to” is considered as an assignment operator of corresponding SAS equivalent syntax.

```
*=====;
* Here is an example of how words such as SET and TO can be processed to sas syntax ;
*=====;
if upcase(split1) ='SET' and upcase(split2)='TO' then do;
  if SPLIT3='' and rev_std_alg='' then sas_code= trim(variable)!!'=' !! substr(STD_alg,index(upcase(STD_alg), '')) );
  type='Internal';
  lib=' ';
end;
*=====;
* The External Type are those the needs data from the external library
*=====;
if upcase(split1) ='SET' and upcase(split2)='TO' and substr(SPLIT3,1,1 ) ne '' and index(STD_ALG, '.') and count( STD_ALG ,'.')=2
and split9= '' then do ;
  data_nm= reverse(scan(reverse(STD_ALG),2));
  lib=SPLIT3;
  sas_code= trim(variable)!!'=' !! reverse(scan(reverse(STD_ALG),1)) ;
  type='External';
end;

*=====;
* The Internal Type derivations does not involve external data source and can be derived within the data step
*=====;

if upcase(split1) ='SET' and upcase(split2)='TO' and substr(SPLIT3,1,1 ) ne '' and index(STD_ALG, '.') and
count( STD_ALG ,'.')=1 and split9= '' then do ;
  data_nm= reverse(scan(reverse(STD_ALG),2));
  lib=' ';
  sas_code= trim(variable)!!'=' !! reverse(scan(reverse(STD_ALG),2))!!'_'!! reverse(scan(reverse(STD_ALG),1)) ;
  type='Internal';
end;
if sas_code ne '';
run;
```

Programming process cont..

Step 3: Additional macro variables are created for each external and internal file references as shown in the programming steps below

```
*%=====;
* Assign macro variables with numbers to each of the External referenced variables
*%=====;
data external/*(keep= type DATA_NM lib )*/;
set spec1 end=end;
count+1;
where type='External';
  call symputx('sasfile'||put(count,4.-1), compress(DATA_NM));
  call symputx('lib'||put(count,4.-1), compress(lib));
  call symputx('code'||put(count,4.-1), trim(SAS_CODE));
  keepvar=substr(SAS_CODE,1,index(SAS_CODE,'=')-1);
  call symputx('keepvar'||put(count,4.-1), trim(keepvar));
  if end then call symputx('maxc',count);

run;

*%=====;
* Assign macro variables with numbers to each of the Internal variables
*%=====;

data Internal/*(keep= type DATA_NM lib )*/;
set spec1 end=end;
count+1;
where type='Internal';
  retain code keepvari;
  if _n_=1 then code=sas_code;
  else if sas_code ne '' then code= trim(code)!!'; '!!trim(sas_code);
  call symputx('code' , trim(code));

  keepvar =substr(SAS_CODE,1,index(SAS_CODE,'=')-1);
  if _n_=1 then keepvari=keepvar;
  else if keepvar ne '' then keepvari= trim(keepvari)!!'; '!!trim(keepvar );
  call symputx('keepvari'||put(count,4.-1), trim(keepvari));
  if end then call symputx('maxi',count);

run;
```

Programming process cont..

Step 4: Ensure that the external datasets that have been referenced exist
SAS Pipe option is used with the filename statement to check the legitimacy of the file path as shown in the code below

```
%put &maxc;
%let byvar= SUBJID;
%let getdata= ;
%let indata=;
*%=====;
* The Fetchdata macro will fetch the data from the corresponding library
*%=====;
%macro fileok;
filename ckfile pipe "dir &&read&i ";
data ckfile ;
    infile ckfile lrecl=200 truncover;
    input file_name_compare $100.;
run;
*%=====;
*If the corresponding dataset is not available in the library then stop the derivation ;
*%=====;
data ckfile;
set ckfile;
    if compress(lowercase(file_name_compare))="dir:cannotaccess" then fileok='N';
    else fileok='Y';
    CALL SYMPUT('Fileok', fileok);
run;
%put &fileok;
%mend fileok;
%macro fetchdata;
```

Programming process cont..

Step 5: : In this final step the %fetchdata macro is been called. The macro is looped based on the number of variables that need to be derived. With each iteration for each variable. The &code&i macro variable will have the corresponding executable syntax using the new variables that are now derived and merged back with the original input data.

```
%macro fetchdata;
  /*=====;
  *The Fetchdata macro will be looped based on the number of variable derivations      ;
  * Each variable will be assigned with a unique macro variable
  /*=====;|
  %do i=1 %to &maxc;
  %fileok;
  %if &fileok=Y %then %do;
    data &&sasfile&i(keep= &byvar &&keepvar&i);
    set &&lib&i...&&sasfile&i ;
    &code&i;
  %let getdata= &&sasfile&i &getdata;
    run;
  proc sort data= &&sasfile&i;
    by &byvar;
  run;
  %end;
  %end;
  data &outdata;
  merge &indata &getdata;
    by &byvar;
    &code;
  run;
  %mend;
%fetchdata;
```

Conclusion

- ❖ Using the standardized language approach, the overall efficiency of SDTM database programming can be improved considerably
- ❖ The future capabilities of the %translate macro will also include the ability to produce categorical and numerical summary statistics based on a one-liner script

Acknowledgements

Sincere thanks to Steve Benjamin, Director Statistical Programming, Biostatistics and Global Contracts for his vision, great leadership, persistent support, and encouragement throughout and for his valuable assistance.

Author's Contact Information

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

Hrideep Antony

Principal Statistical Programmer, Clinical Division, Syneos Health

Phone: +1 984 459 4785

E-mail: Hrideep.Antony@syneoshealth.com

Web: <http://www.syneoshealth.com>

Aman Bahl

Associate Director, Statistical Programming, Clinical Division, Syneos Health

Phone: +1 905 399 6715

E-mail: Aman.Bahl@syneoshealth.com

Web: <http://www.syneoshealth.com>

