PhUSE US Connect 2018

Paper SI07

Optimizing SDTM Specification Development with Auto-population

Cori Kramer, Covance Inc., King of Prussia, PA
Ryan Adalbert, Covance Inc., King of Prussia, PA

ABSTRACT

On many studies, the development of SDTM specifications can be optimized using SAS macros. During the creation of the SDTM data, the author manually identifies mapping rules and a certain level of metadata, for which CDISC review software or sponsor created specs can be used. When the raw data has several levels of SDTM compliance, SAS macros can be utilized to compare raw data values to the CDISC Controlled Terminology database. Macros can also check the conformance of raw values and metadata with SDTM standards. This enables the auto-population of mapping rules into the specification, significantly reducing the manual work time.

INTRODUCTION

Providing specifications for programmers to use when creating SDTM dataset conversion programs is an essential part of the SDTM process. Creating these specifications frequently involves excessive manual work that may be repetitive in nature, especially when raw data has substantial SDTM compliance. Fortunately, the utilization of SAS macros during the spec writing process offers significant benefits as macros can:

- Provide high level overviews of domains or raw datasets.
- Ensure that all raw variables are mapped.
- Dramatically increase the speed of producing programming notes by auto-populating case statements and commonly used notes.
- Eliminate the need to check all raw values against individual CDISC Controlled Terminology (CT) by instantly identifying conformant and non-conformant values.
- Ensure that certain variables such as USUBJID, STUDYID, and date/time variables conform to their expected formats.
- Allow the specification to adopt a uniform structure as auto-population results in mapping comments, wording, and casing that is the same throughout.

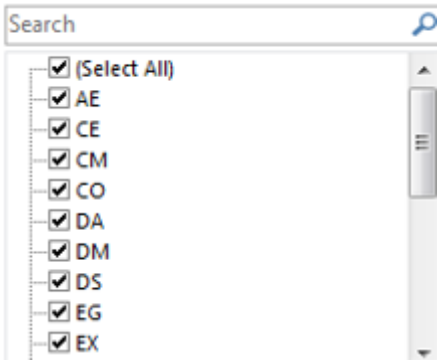
These benefits can considerably reduce the required work time of the spec author and produce higher quality, and less error prone specifications. Furthermore, these benefits can be achieved by using a single SAS macro that leverages MS Excel such that the spec author has less of a need to directly review raw data sets or terminology files.

UNDERSTAND RAW DATA

OVERVIEW OF RAW DATASETS AND VARIABLES

When spec authors begin work on a new study, they must first develop an understanding of the layout of the data and the raw variables present in each dataset. To assist in this process, SAS macros can be utilized to compile details of the variables from each raw dataset in a user friendly excel spreadsheet. This allows the spec author to simultaneously view and filter through the raw variables, values, and certain metadata from dozens of datasets. The spec author can use excel to filter the information by raw dataset and can gain an overview of the structure and where changes may need to occur. The example below demonstrates this stage of the macro having been run on several raw SDTM-like domains.

Raw_Dataset	Raw_Variable	Raw_Dataset	Raw_Variable
AE	AEPATT	AE	AESTDTC
AE	AEREL	CE	CEDTC
AE	AESCONG	CE	CEENDTC
AE	AESDISAB	CE	CESTDTC
AE	AESDTH	CM	CMENDTC
AE	AESEQ	CM	CMSTDTC
AE	AESER	CO	CODTC
AE	AESEV	DA	DADTC
AE	AESHOSP	DM	BRTHDTC
AE	AESLIFE	DM	DMDTC
AE	AESMIE	DM	DTHDTC
AE	AESPID	DM	RFENDTC



A quick glance at the Excel spreadsheet output by the macro shows raw dataset names and fields very similar to SDTM. This type of data is where the optimization power of the macro is maximized. One of the biggest benefits of this type of view is the ability to utilize the sort and filter features of Excel, which can be used to look at individual domains or at similar variables across multiple domains.

NO RAW VARIABLE LEFT BEHIND

Every raw variable needs to be assigned mapping instructions which helps to prevent raw variables from being overlooked or missed during mapping.

CODING

The raw variables from each domain can be captured using PROC CONTENTS procedures as each raw dataset is brought in. It is only necessary to keep the variables MEMNAME and NAME which store the dataset name and variable names respectively.

```
proc contents data=&d1
       out=&d1.vars1(keep=memname name);
run;
```

Next, the results of the PROC CONTENTS procedures can be consolidated into a single dataset. MEMNAME should be renamed to DOMAIN and NAME should be renamed to VARIABLE. The dataset should then be sorted by DOMAIN and VARIABLE.

```
data rawcontents(rename=(memname=Domain name=Variable));
       set &d1._vars1;
proc sort;
       by domain variable;
run;
```

The consolidated dataset should then be included as one of the datasets in the first PROC EXPORT procedure.

```
proc export data=rawcontents
   dbms=XLSX
       outfile='Desired location' replace;
              sheet='Raws';
```

CHECK CONTROLLED TERMINOLOGY

CHECKING RAW VALUES AGAINST CDISC CT

After a high-level understanding of the data has been established, the spec author must then dive into examining the raw variable values. In the same spreadsheet, the spec author can then view unique raw values next to the appropriate Controlled Terminology. Sponsor CT can be added where necessary and values that do not directly match up to an existing CT value can be matched. This allows the spec author to see which values are compliant. The spec author can also enter custom mapping instructions for each value that they wish to have populated in the spec. The example below features this stage of the macro on the raw Vital Signs domain.

Raw_Dataset	Raw_Variable	Uniques	Code	Codelist_Code	Codelist_Nam	CDISC_Submission_Value	CDISC_Synonyn
VS	VSPOS	NOT DONE (AFTER 5 MIN REST)				null	
VS	VSPOS	SITTING (AFTER 5 MIN REST)				SITTING	
VS	VSPOS	STANDING (AFTER 5 MIN REST)				STANDING	
VS	VSPOS	SUPINE (AFTER 5 MIN REST)				SUPINE	
VS	VSPOS	SITTING	C62122	C71148	Position	SITTING	Sitting
VS	VSPOS	STANDING	C62166	C71148	Position	STANDING	Standing
VS	VSPOS	SUPINE	C62167	C71148	Position	SUPINE	Supine

The Excel sheet shown above lists the unique variable values that match identically with CDISC controlled terminology. Controlled Terminology values can also be entered by the author to indicate that the macro should write if-then statements standardizing these values.

CODING

First, a PROC IMPORT procedure should be used to read in the relevant CDISC Controlled Terminology spreadsheet. The SHEET= option must be used to indicate which tab in the excel sheet should be used. This can be accomplished by setting a macro variable as shown in the example below. Additionally, the GETNAMES option was not used due to spacing issues with the column names in the Excel sheet.

```
filename term '/location/SDTM Terminology.xls';
proc import datafile=term replace

dbms=XLS
       out=term1;
              sheet = &sheet;
                     getnames=NO;
                            datarow=2;
   run;
```

The link between the raw variable values and the Controlled Terminology sheet is the “CDISC Submission Value” column. Therefore, the CDISC Submission Value (column E) is renamed to RAWVALUE in preparation for a merge with the raw datasets.

```
data term2;
       set term1;
              rename E = RawValue;
proc sort;
       by RawValue;
run;
```

Next, PROC TRANSPOSE should be used to transpose the domain names, variables, and raw values by all the variables present in the datasets.

```
proc transpose data=&d1 out=data2(keep=domain _name_ col1);
       by _all_;
              var _all_;
run;
```

After the datasets are transposed, only the unique observations should be kept. RAWVALUE is the variable which stores variable values.

```
proc sql;
	create table u1 as select distinct
		domain as Domain, _name_ as Variable, col1 as RawValue, col1 as Uniques
from data2
order by RawValue;
quit;
```

Next, the transposed raw datasets and the Controlled Terminology dataset should all be merged together with the variable RAWVALUE as the link between all datasets.

```
data final;
	merge term2 u1;
		by RawValue;
proc sort;
	by A;
run;
```

Finally, after variable names have been corrected and the dataset is sorted, the combined dataset should be output to Excel using the PROC EXPORT procedure.

```
proc export data=final
	dbms=XLSX
		outfile='/location/Spreadsheet'
			replace;
run;
```

CHECK RAW VALUES FOR CONFORMANCE

CHECKING VARIABLE FORMATS

Using the Excel sheet produced by the macro, the spec author can then check other variables for SDTM conformance that do not contain controlled terminology. For example, the Excel screenshot below shows the unique date format of VSDTC. The format displayed under the UNIQUES column shows that the variable VSDTC is compliant with CDISC standards. In addition, fields such as study ID, subject ID, record identifiers, etc. can be analyzed, and their formatting displayed.

Raw_Dataset	Raw_Variable	Uniques
VS	VSDTC	yyyy-mm-dd

CODING

The first step is to determine which variables should be checked for compliance. Once the necessary variables have been chosen, different combinations of the INDEX, SCAN, SUBSTR, and LENGTH functions can be used to test and report the formatting of the variable. In the example below, the SUBSTR function is used to test whether VSDTC conforms to ISO8601 YYYY-MMM-DD format.

```
&d1.dtc = strip(&d1.dtc);
	if substr(&d1.dtc,5,1) = '-'
		and substr(&d1.dtc,8,1) = '-'
			and length(&d1.dtc) = 10
				then &d1.dtc = 'yyyy-mm-dd';
```

Similar statements can be added to test for other formats and additional tests can be added for other variables. Because only unique values are output to the Excel spreadsheet, a single observation will display the format of the variable as shown above. If a variable has multiple detected formats, then each unique format will be displayed in the spreadsheet.

POPULATE SPECIFICATIONS

SPECIFYING MAPPING INSTRUCTIONS

Next, the spec author indicates which variables they want to map and how the mapping should be performed. Single character identifiers will be used to specify which pre-programmed notes should output in the specification. Some examples could be identifying where dates need to be changed to ISO8601 format, identifying where values need to be updated to match CDISC CT, or simply straight mapping variables from raw to SDTM where they already conform to the proper SDTM values and format.

For example, when variables are already SDTM compliant, the author can then enter a Y to indicate that the date can be mapped as is. In this case, VSDTC will be entered into the programming notes column of the spec with text indicating that the variable should be straight mapped. The example below demonstrates this stage of the macro on the raw Vital Signs domain.

Raw_Dataset	Raw_Variable	Mapping_Instruction
VS	VSDTC	Y

GENERATING SPECIFICATIONS

The macro then reads in the single character identifiers alongside the Controlled Terminology comparison to output the mapping notes in the specification. Next, the macro auto-populates the specification with the appropriate programming notes based on the user's instructions. Lastly, a high-level check of the raw data can be done to ensure no structural issues have been missed and to populate any manual notes that are needed.

Dataset	Variable	Programming_Notes
		When VS.vspos = 'NOT DONE (AFTER 5 MIN REST)' set to missing
		When VS.vspos = 'SITTING (AFTER 5 MIN REST)' set to 'SITTING'
		When VS.vspos = 'STANDING (AFTER 5 MIN REST)' set to 'STANDING'
		When VS.vspos = 'SUPINE (AFTER 5 MIN REST)' set to 'SUPINE'
VS	VSPOS	Otherwise VS.vspos

In the example above, the programming and mapping notes are output in the specification when the macro is run. In this case, the if-then statement mentioned previously is output. The manually entered matches are in the when conditions of the if-then statement and the remaining values are mapped as is.

UNIFORM SPECIFICATIONS

One of the benefits of using this approach is that the specifications will have a uniform feel as notes and mapping instructions will be written the same across all specs.

CODING

First, the specification template and the modified Excel spreadsheet must be imported using the PROC IMPORT procedure. Next, the programming instructions column of the specification will be populated based on the input from the spec author. Pre-programmed conditions generate the appropriate output. In the example from this section, the spec author specified a Y, which will result in a straight mapping of the given variable.

```
if upcase(Straight_Map) = 'Y' then programming_notes = strip(domain) || '.' ||
strip(lowcase(variable));
```

If the spec author decides not to straight map a variable, then the code shown below will output mapping instructions to the spec that inform the programmer to set the variable to a controlled terminology value when a certain raw value is present.

```
if upcase(Straight_Map) = 'N' then do;

if cdisc_submission_value ne uniques then programming_notes = 'When ' ||
strip(domain) || '.' || strip(lowcase(variable)) || ' = ' || "'" || strip(uniques) ||
"'"|| ' set to ' || "'"|| strip(cdisc_submission_value) ||"'     " ;

end;
```

Designated key terms can be used to write specific mapping instructions to the specification template. For example, placing a Y in the Manual Map column of the spreadsheet will print a note saying “MANUAL MAPPING REQUIRED” to the programming notes column of the specification. This note serves as a reminder to the spec author that manual instructions must be written in this instance. Additionally, placing an M in the Manual Map column will print the note “Set to null.” Other key terms can also be specified as shown below.

```
if upcase(Manual_Map) = 'Y' then programming_notes = 'MANUAL MAPPING REQUIRED';

if upcase(Manual_Map) = 'M' then programming_notes = 'Set to null';

if upcase(Manual_Map) = 'V' then programming_notes = 'Assign using' || strip(domain)
|| '.visit,' || strip(domain) || '.visitnum and VISIT_LUT';

if upcase(Manual_Map) = 'U' then programming_notes = 'Uppercase ' || strip(domain) ||
'.' || strip(lowcase(variable));

if upcase(Manual_Map) = 'E' then programming_notes = 'Set to SE.EPOCH where
SE.SESTDTC <= (xx.xxSTDTC, or xxDTC for findings domains) <= SE.SEENDTC';
```

Lastly, the programming instructions variable is merged into the dataset containing the specification template and the final dataset is sorted by the appropriate keys. The specification is then published using the PROC EXPORT procedure.

CONCLUSION

Creating SDTM specifications for guiding the development of dataset conversion programs is an essential part of the SDTM process. Fortunately, utilizing SAS macros as discussed can eliminate most of the manual and repetitive work from the spec writing process. SAS macros save time and effort by eliminating the need to check all raw values against CDISC Controlled Terminology and by auto-populating the specification template. SAS macros also provide a data overview, ensure that raw variables are mapped, ensure that variables conform to their expected format, and enforce a uniform structure throughout the spec. This process produces more efficient, higher quality, and less error prone specifications.

CONTACT INFORMATION

Cori Kramer
Covance
1016 West Ninth Avenue Suite 300
King of Prussia, PA 19406
Work Phone: 484-942-4844
Email: cori.kramer@chiltern.com

Ryan Adalbert
Covance
1016 West Ninth Avenue Suite 300
King of Prussia, PA 19406
Work Phone: 484-679-2541
Email: ryan.adalbert@chiltern.com