PharmaSUG 2013 - Paper DS03

Programming Validation Tips for SDTM prior to using OpenCDISC validator

Dany Guerendo, STATProg LLC, Morrisville, NC

ABSTRACT:

In the years I have been working with the Clinical Data and Standards Consortium (CDISC), primarily using the standard data tabulation model (SDTM), validating the domains I create can prove to be challenging for programmers new to the standards. This paper provides tips and techniques, developed for validating domains created; prior to running into a validation tool like OpenCDISC or WebSDM.

This paper‘s sole purpose is to help facilitate the task of the primary programmer or the validation programmer, if applicable, by automating some of the repetitive tasks occurring when programming SDTM.

It may not be an exhaustive list of options but I hope it serves as guidance document for programmers.

Although I worked mostly with version 3.1.2 of the SDTM Implementation Guide, these tips work well with version 3.1.3 which is now available.

These programming tips were applied in SAS interactive (Window based version) version 9.2 and 9.3 as well as SAS Enterprise Guide version 3.1 and 5.3.

INTRODUCTION:

This paper will describe a method for automating the assignment of SDTM domain labels, for variables as well as datasets.

It will also provide ways to assign controlled terminology, or check that the controlled terminology is properly applied.

Lastly, it will show how to determine where a difference in observation counts occurred when validating rather large findings observation class in SDTM.

SDTM DOMAIN: VARIABLE LABELS AND DOMAIN LABEL

One simple solution to the frequently encountered issue of mistyped variable labels is to automate the process. Some companies may already have programs or software implemented to handle this; but if you do not, you can create a dataset from the SDTM IG excel spreadsheet listing all domains, also available online by following the links to SDTM standards on the CDISC website: www. CDISC.org. This spreadsheet is available for version 3.1.2 but may not be available for version 3.13. I reformatted the spreadsheet to look like this but it is a personal preference:

Below is a sample SDTM metadata spreadsheet.

Version	Dlabel	Domain	Vname	Vlabel	Vtype	Core	Vorder	Indomain
v3.1.3	Demographics	DM	STUDYID	Study Identifier	Char	Req	1	D
v3.1.3	Demographics	DM	DOMAIN	Domain Abbreviation	Char	Req	2	D
v3.1.3	Demographics	DM	USUBJID	Unique Subject Identifier	Char	Req	3	D
v3.1.3	Demographics	DM	SUBJID	Subject Identifier for the Study	Char	Req	4	D
v3.1.3	Demographics	DM	RFSTDTC	Subject Reference Start Date/Time	Char	Exp	5	D
v3.1.3	Demographics	DM	RFENDTC	Subject Reference End Date/Time	Char	Exp	6	D
v3.1.3	Demographics	DM	RFXENDTC	Date/Time of First Study Treatment	Char	Exp	7	D
v3.1.3	Demographics	DM	RFXSTDTC	Date/Time of Last Study Treatment	Char	Exp	8	D
v3.1.3	Demographics	DM	RFICDTC	Date/Time of Informed Consent	Char	Exp	9	D

v3.1.3	Demographics	DM	RFPENDTC	Date/Time of End of Participation	Char	Exp	10	D
v3.1.3	Demographics	DM	DTHDTC	Date/Time of Death	Char	Exp	11	D
v3.1.3	Demographics	DM	DTHFL	Subject Death Flag	Char	Exp	12	D
v3.1.3	Demographics	DM	SITEID	Study Site Identifier	Char	Req	13	D
v3.1.3	Demographics	DM	INVID	Investigator Identifier	Char	Perm	14	D
v3.1.3	Demographics	DM	INVNAM	Investigator Name	Char	Perm	15	D
v3.1.3	Demographics	DM	BRTHDTC	Date/Time of Birth	Char	Perm	16	D
v3.1.3	Demographics	DM	AGE	Age	Num	Exp	17	D
v3.1.3	Demographics	DM	AGEU	Age Units	Char	Exp	18	D

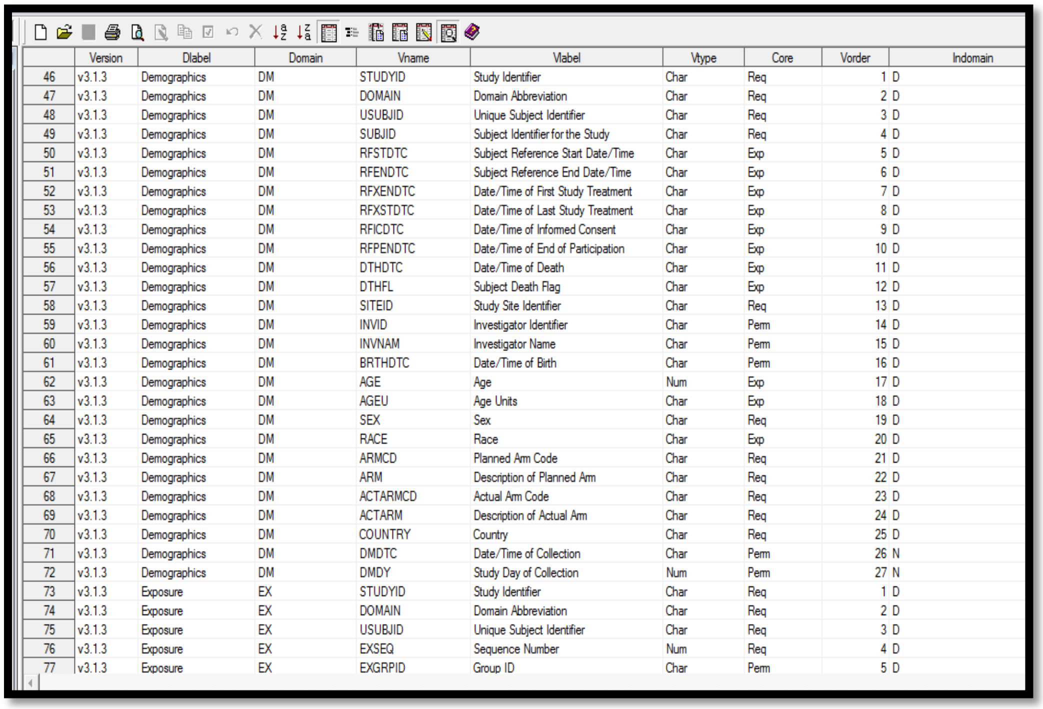
The last column is used for the sole purpose of selecting the variables that are created in the domain. This allows you to select only the variables that are part of the domain programmed since is possible that some permissible variables (Core =Perm) are not used.
The display order of the variables is key in the SDTM guidelines so insuring that the order is maintained should be a part of any macros automating the assignment of variable attributes.

From this spreadsheet we create a SAS dataset. Below is an example of a simple SAS macro to read in the excel spreadsheet you created.

```
%macro sdtm();

filename sdtmct "xxxxxxx/cdisc_v_312.csv";

*-------------------------------------------------------------------*
*     Read in excel spreadsheet
*     Output permanent dataset with all domains attributes
*-------------------------------------------------------------------*;
proc import out= sdtmqc.cdisc_v_312
      datafile = sdtmct
      dbms=csv replace;
      getnames=yes;
run;

filename sdtmct clear;

%mend;
```

Once executed this program creates a readily available dataset of all SDTM variable labels.
See an example of the output below:

Sample SDTM attribute dataset.



	Version	Dlabel	Domain	Vname	Vlabel	Vtype	Core	Vorder	Indomain
46	v3.1.3	Demographics	DM	STUDYID	Study Identifier	Char	Req	1	D
47	v3.1.3	Demographics	DM	DOMAIN	Domain Abbreviation	Char	Req	2	D
48	v3.1.3	Demographics	DM	USUBJID	Unique Subject Identifier	Char	Req	3	D
49	v3.1.3	Demographics	DM	SUBJID	Subject Identifier for the Study	Char	Req	4	D
50	v3.1.3	Demographics	DM	RFSTDTC	Subject Reference Start Date/Time	Char	Exp	5	D
51	v3.1.3	Demographics	DM	RFENDTC	Subject Reference End Date/Time	Char	Exp	6	D
52	v3.1.3	Demographics	DM	RFXENDTC	Date/Time of First Study Treatment	Char	Exp	7	D
53	v3.1.3	Demographics	DM	RFXSTDTC	Date/Time of Last Study Treatment	Char	Exp	8	D
54	v3.1.3	Demographics	DM	RFICDTC	Date/Time of Informed Consent	Char	Exp	9	D
55	v3.1.3	Demographics	DM	RFPENDTC	Date/Time of End of Participation	Char	Exp	10	D
56	v3.1.3	Demographics	DM	DTHDTC	Date/Time of Death	Char	Exp	11	D
57	v3.1.3	Demographics	DM	DTHFL	Subject Death Flag	Char	Exp	12	D
58	v3.1.3	Demographics	DM	SITEID	Study Site Identifier	Char	Req	13	D
59	v3.1.3	Demographics	DM	INVID	Investigator Identifier	Char	Perm	14	D
60	v3.1.3	Demographics	DM	INVNAM	Investigator Name	Char	Perm	15	D
61	v3.1.3	Demographics	DM	BRTHDTC	Date/Time of Birth	Char	Perm	16	D
62	v3.1.3	Demographics	DM	AGE	Age	Num	Exp	17	D
63	v3.1.3	Demographics	DM	AGEU	Age Units	Char	Exp	18	D
64	v3.1.3	Demographics	DM	SEX	Sex	Char	Req	19	D
65	v3.1.3	Demographics	DM	RACE	Race	Char	Exp	20	D
66	v3.1.3	Demographics	DM	ARMCD	Planned Arm Code	Char	Req	21	D
67	v3.1.3	Demographics	DM	ARM	Description of Planned Arm	Char	Req	22	D
68	v3.1.3	Demographics	DM	ACTARMCD	Actual Arm Code	Char	Req	23	D
69	v3.1.3	Demographics	DM	ACTARM	Description of Actual Arm	Char	Req	24	D
70	v3.1.3	Demographics	DM	COUNTRY	Country	Char	Req	25	D
71	v3.1.3	Demographics	DM	DMDTC	Date/Time of Collection	Char	Perm	26	N
72	v3.1.3	Demographics	DM	DMDY	Study Day of Collection	Num	Perm	27	N
73	v3.1.3	Exposure	EX	STUDYID	Study Identifier	Char	Req	1	D
74	v3.1.3	Exposure	EX	DOMAIN	Domain Abbreviation	Char	Req	2	D
75	v3.1.3	Exposure	EX	USUBJID	Unique Subject Identifier	Char	Req	3	D
76	v3.1.3	Exposure	EX	EXSEQ	Sequence Number	Num	Req	4	D
77	v3.1.3	Exposure	EX	EXGRPID	Group ID	Char	Perm	5	D

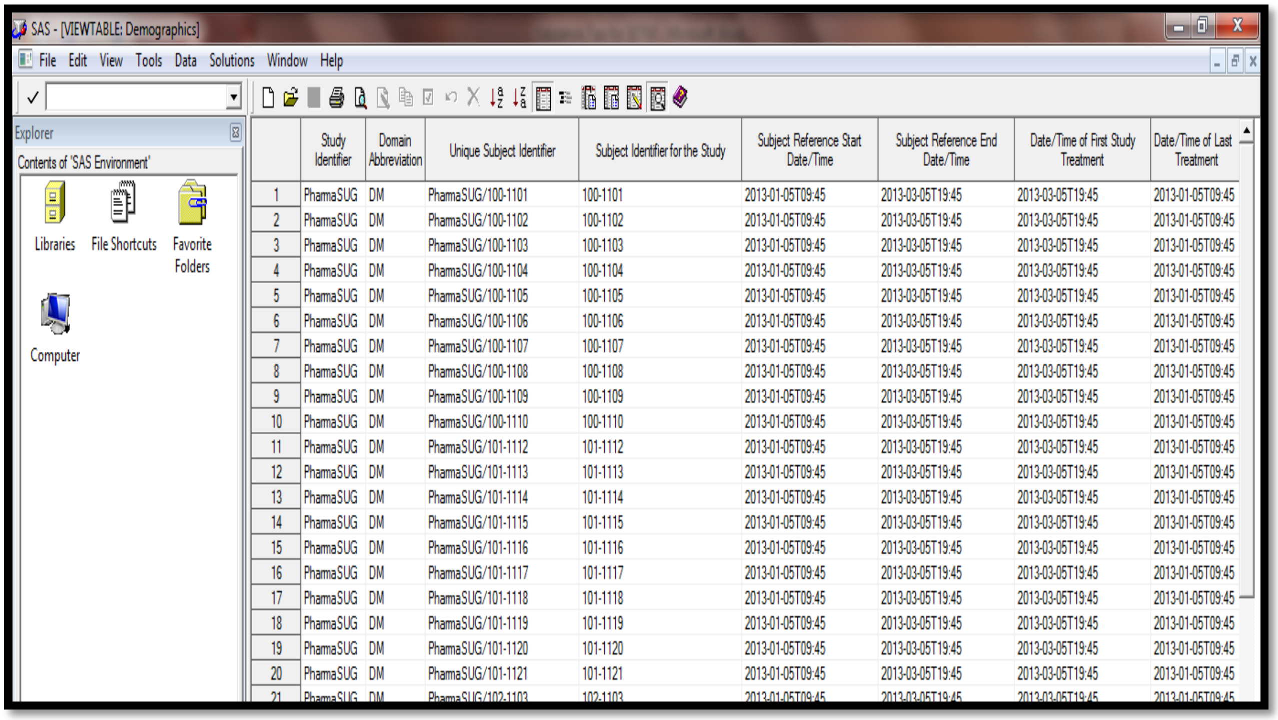
Version: The variable allows you to enter several versions of the domain in the spreadsheet.
Domain: The variable selecting which domain attributes you need in the run.
Dlabel: All SDTM dataset labels
Now having access to this data, a macro can be created to assign variable and dataset labels.
The macro below is a fairly simple example of how you can accomplish this task.

```
*---------------------------------------------------------------------*
*        Select a the appropriate domain from dataset CDISC_v_312
*        Create a permanent dataset
*        with the appropriate attributes from the input work dataset
*--------------------------------------------------------------------*;
%macro attribut(dsin=, libsdtm =, domain =, dssort =);

data attrib(where=(compress(upcase(domain))="&domain"));
      set &libsdtm..cdisc_v_312(keep =version dlabel domain vname
vlabel vtype core vorder indomain);
      if compress(indomain) =:'N' then delete;
run;

proc sort data =attrib;
      by vname;
run;

*---------------------------------------------------------------------*
*Check content of created domain: variable names, labels, type and
*length
*--------------------------------------------------------------------*;
proc sql noprint;
      create table content as
      select libname, upcase(name) as vname length=8, label as wklabel,
      type as wktype label ="Type as defined in &dsin", length as
wklength
```

```
      from dictionary.columns
      where libname ='WORK' and memname =upcase("&dsin")
      order by vname;
quit;

*-----------------------------------------------------------------------*
*Merge content of created domain with attrib
*Check attributes in work dataset against attrib
*----------------------------------------------------------------------*;
data attrib2;
merge attrib(in=at) content(in=ct);
      by vname;
** Create length of variables based on work dataset attributes;
if upcase(vtype) ='CHAR' then clength ='$'||strip(put(wklength,best.));
            else clength =strip(put(wklength,best.));

if at and substr(vname,3) ='SEQ' then clength =strip(put(8,best.));
if at and vname eq 'DOMAIN' then clength ='$'||strip(put(2,best.));
      if at then output attrib2;

run;

*-----------------------------------------------------------------------*
        Create macro variable of record counts in attribute dataset
*----------------------------------------------------------------------*;
proc sql noprint;
      select count(*) into: obscnt
      from attrib2(where=(domain="&domain"))
      ;
quit;

%let obs =%sysfunc(compress(&obscnt));

proc sort data =attrib2;
    by vorder;
run;

*-----------------------------------------------------------------------*
*     Create macro variables of variable names, labels, type and length
*----------------------------------------------------------------------*;
proc sql noprint;
    select vname into: varnm1-:varnm&obs
    from attrib2(where=(indomain ='D'))
    ;
    select vlabel into: lbl1-:lbl&obs
    from attrib2(where=(indomain ='D'))
    ;
    select vtype into: type1-:type&obs
    from attrib2(where=(indomain ='D'))
    ;
    select clength into: lgth1-:lgth&obs
    from attrib2(where=(indomain ='D'))
    ;
quit;
```

```
*---------------------------------------------------------------------*
*     Create dataset label
*--------------------------------------------------------------------*;
proc sql noprint;
    select strip(dlabel) into: dslabel
    from attrib(where=(upcase(domain)="&domain"))
    ;
quit;

*---------------------------------------------------------------------*
        Create full attribute text
*--------------------------------------------------------------------*;
data _null_;
    %do i=1 %to &obs;
      call symput("attrib" ||strip(put(&i,best.)), "&&varnm&i" || "
label=" || "'&&lbl&i'" || " length=" || "&&lgth&i" || " format=" ||
"&&lgth&i"||".");
    %end;
run;

*---------------------------------------------------------------------*
        Output SDTM dataset with proper variable attributes
*--------------------------------------------------------------------*;
data &domain(label ="&dslabel"
                keep =
                %do i=1 %to &obs;
                        &&varnm&i
                %end;) &dsin._;
        attrib
        %do i=1 %to &obs;
                &&attrib&i
        %end;
        ;
        set &dsin;
        by &dssort;

        ** Assign domain name;
        domain =compress("&domain");
run;

%mend attribut;

**Sample call;
*%attribut(dsin=, libsdtm =, domain =, dssort =);
```

An example of a demographic domain created using the macro ATTRIBUT.



	Study Identifier	Domain Abbreviation	Unique Subject Identifier	Subject Identifier for the Study	Subject Reference Start Date/Time	Subject Reference End Date/Time	Date/Time of First Study Treatment	Date/Time of Last Treatment
1	PhamaSUG	DM	PhamaSUG/100-1101	100-1101	2013-01-05T09:45	2013-03-05T19:45	2013-03-05T19:45	2013-01-05T09:45
2	PhamaSUG	DM	PhamaSUG/100-1102	100-1102	2013-01-05T09:45	2013-03-05T19:45	2013-03-05T19:45	2013-01-05T09:45
3	PhamaSUG	DM	PhamaSUG/100-1103	100-1103	2013-01-05T09:45	2013-03-05T19:45	2013-03-05T19:45	2013-01-05T09:45
4	PhamaSUG	DM	PhamaSUG/100-1104	100-1104	2013-01-05T09:45	2013-03-05T19:45	2013-03-05T19:45	2013-01-05T09:45
5	PhamaSUG	DM	PhamaSUG/100-1105	100-1105	2013-01-05T09:45	2013-03-05T19:45	2013-03-05T19:45	2013-01-05T09:45
6	PhamaSUG	DM	PhamaSUG/100-1106	100-1106	2013-01-05T09:45	2013-03-05T19:45	2013-03-05T19:45	2013-01-05T09:45
7	PhamaSUG	DM	PhamaSUG/100-1107	100-1107	2013-01-05T09:45	2013-03-05T19:45	2013-03-05T19:45	2013-01-05T09:45
8	PhamaSUG	DM	PhamaSUG/100-1108	100-1108	2013-01-05T09:45	2013-03-05T19:45	2013-03-05T19:45	2013-01-05T09:45
9	PhamaSUG	DM	PhamaSUG/100-1109	100-1109	2013-01-05T09:45	2013-03-05T19:45	2013-03-05T19:45	2013-01-05T09:45
10	PhamaSUG	DM	PhamaSUG/100-1110	100-1110	2013-01-05T09:45	2013-03-05T19:45	2013-03-05T19:45	2013-01-05T09:45
11	PhamaSUG	DM	PhamaSUG/101-1112	101-1112	2013-01-05T09:45	2013-03-05T19:45	2013-03-05T19:45	2013-01-05T09:45
12	PhamaSUG	DM	PhamaSUG/101-1113	101-1113	2013-01-05T09:45	2013-03-05T19:45	2013-03-05T19:45	2013-01-05T09:45
13	PhamaSUG	DM	PhamaSUG/101-1114	101-1114	2013-01-05T09:45	2013-03-05T19:45	2013-03-05T19:45	2013-01-05T09:45
14	PhamaSUG	DM	PhamaSUG/101-1115	101-1115	2013-01-05T09:45	2013-03-05T19:45	2013-03-05T19:45	2013-01-05T09:45
15	PhamaSUG	DM	PhamaSUG/101-1116	101-1116	2013-01-05T09:45	2013-03-05T19:45	2013-03-05T19:45	2013-01-05T09:45
16	PhamaSUG	DM	PhamaSUG/101-1117	101-1117	2013-01-05T09:45	2013-03-05T19:45	2013-03-05T19:45	2013-01-05T09:45
17	PhamaSUG	DM	PhamaSUG/101-1118	101-1118	2013-01-05T09:45	2013-03-05T19:45	2013-03-05T19:45	2013-01-05T09:45
18	PhamaSUG	DM	PhamaSUG/101-1119	101-1119	2013-01-05T09:45	2013-03-05T19:45	2013-03-05T19:45	2013-01-05T09:45
19	PhamaSUG	DM	PhamaSUG/101-1120	101-1120	2013-01-05T09:45	2013-03-05T19:45	2013-03-05T19:45	2013-01-05T09:45
20	PhamaSUG	DM	PhamaSUG/101-1121	101-1121	2013-01-05T09:45	2013-03-05T19:45	2013-03-05T19:45	2013-01-05T09:45
21	PhamaSUG	DM	PhamaSUG/102-1103	102-1103	2013-01-05T09:45	2013-03-05T19:45	2013-03-05T19:45	2013-01-05T09:45

SAS - [VIEWTABLE: Demographics]
File Edit View Tools Data Solutions Window Help
Explorer
Contents of 'SAS Environment'
Libraries
File Shortcuts
Favorite Folders
Computer

QUALITY CHECK ON CONTROLLED TERMINOLOGY

One of the most common issue we encounter is, when controlled terminology is applied, how do we recognized terms that do not comply?

One idea is to create a format catalog from the controlled terminology(CT) spreadsheet available from the National Cancer Institute website; looks like this:

This is a snapshot of the CDISC controlled terminology.

Code	Codelist Code	Codelist Extensible (Yes/No)	Codelist	Codelist Name	CDISC Submission Value	CDISC Synonym(s)
C66767		No	ACN	Action Taken with Study Treatment	ACN	Action Taken with Study Treatment
C49503	C66767		ACN	Action Taken with Study Treatment	DOSE INCREASED	
C49504	C66767		ACN	Action Taken with Study Treatment	DOSE NOT CHANGED	
C49505	C66767		ACN	Action Taken with Study Treatment	DOSE REDUCED	
C49501	C66767		ACN	Action Taken with Study Treatment	DRUG INTERRUPTED	
C49502	C66767		ACN	Action Taken with Study Treatment	DRUG WITHDRAWN	
C48660	C66767		ACN	Action Taken with Study Treatment	NOT APPLICABLE	NA

C17998	C66767		ACN	Action Taken with Study Treatment	UNKNOWN	U; Unknown
C66768		No	OUT	Outcome of Event	OUT	Outcome of Event

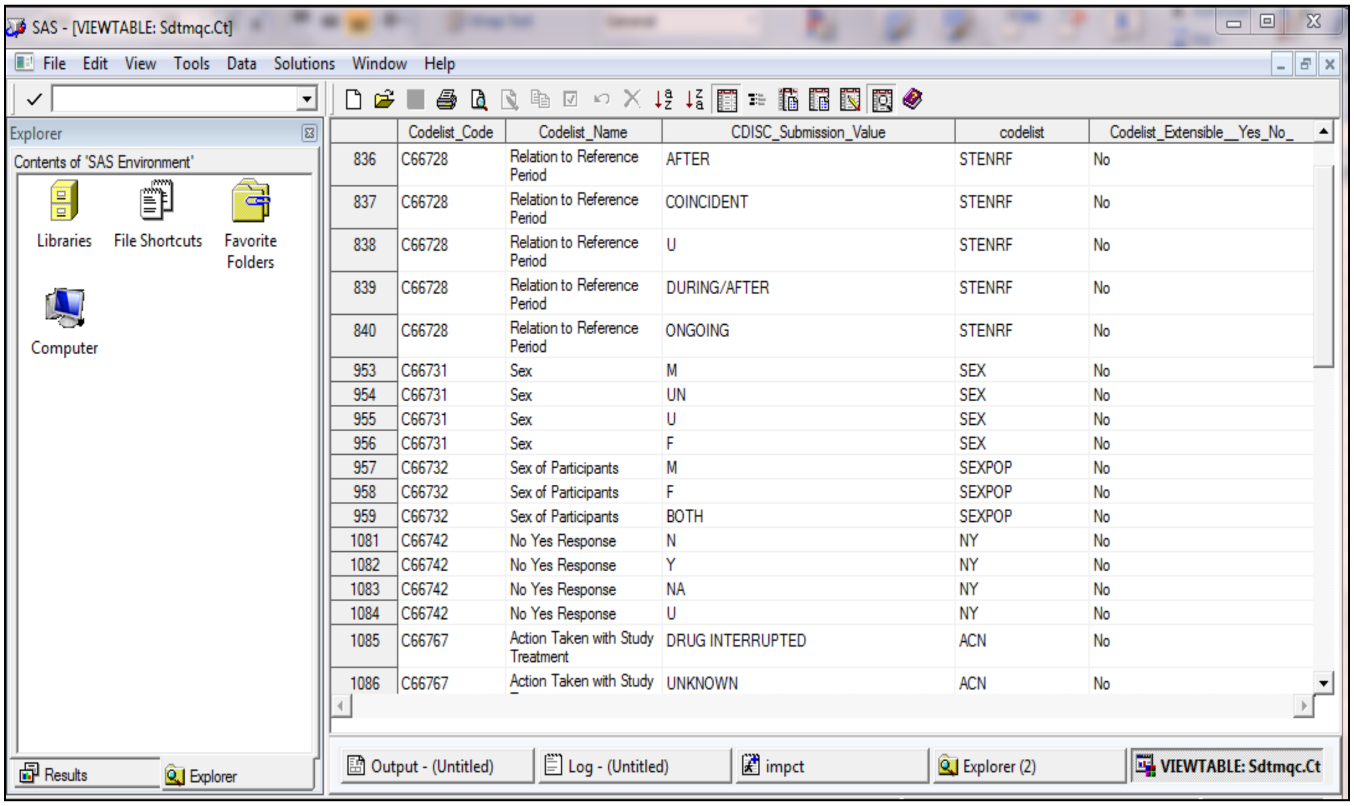
ACN is the SDTM code list applied to the AE domain variable, AEACN for action taken.

First read the controlled terminology in SAS as a dataset using proc import or data steps.

```
filename sdtmct "xxxxxxxxxxxx/SDTM Terminology.csv";

proc import out= ctemp1
      datafile = sdtmct
      dbms=csv replace;
      getnames=yes;
run;

filename sdtmct clear;
```

Create a SAS dataset from the imported controlled terminology.

Table below shows an example of resulting data:



	Codelist_Code	Codelist_Name	CDISC_Submission_Value	codelist	Codelist_Extensible__Yes_No_
836	C66728	Relation to Reference Period	AFTER	STENRF	No
837	C66728	Relation to Reference Period	COINCIDENT	STENRF	No
838	C66728	Relation to Reference Period	U	STENRF	No
839	C66728	Relation to Reference Period	DURING/AFTER	STENRF	No
840	C66728	Relation to Reference Period	ONGOING	STENRF	No
953	C66731	Sex	M	SEX	No
954	C66731	Sex	UN	SEX	No
955	C66731	Sex	U	SEX	No
956	C66731	Sex	F	SEX	No
957	C66732	Sex of Participants	M	SEXPOP	No
958	C66732	Sex of Participants	F	SEXPOP	No
959	C66732	Sex of Participants	BOTH	SEXPOP	No
1081	C66742	No Yes Response	N	NY	No
1082	C66742	No Yes Response	Y	NY	No
1083	C66742	No Yes Response	NA	NY	No
1084	C66742	No Yes Response	U	NY	No
1085	C66767	Action Taken with Study Treatment	DRUG INTERRUPTED	ACN	No
1086	C66767	Action Taken with Study	UNKNOWN	ACN	No

The dataset was manipulated to populate the "code list extensible" variable for all rows.

Keeping this column, indicating whether a code list is extensible or not, could prove very practical for people new to the standard. All the code lists appearing in the table above are non-extensible. This means you do **not** have the flexibility to deviate from what CDISC proposes for that code list.

RACE is an extensible code list. If you consistently collect a race category that does not appear in the CDISC terminology, you may suggest the value be added. The code list is updated at least once a year so it is a good idea to check frequently if a new one is available as new terms may be added.

There are, I am sure, many ways to use this data to automate assignment of a format to a variable.

If access to the database code list is available, one way would be to create a dataset with a column for all values in your raw file and the corresponding CDISC controlled terminology assigned.

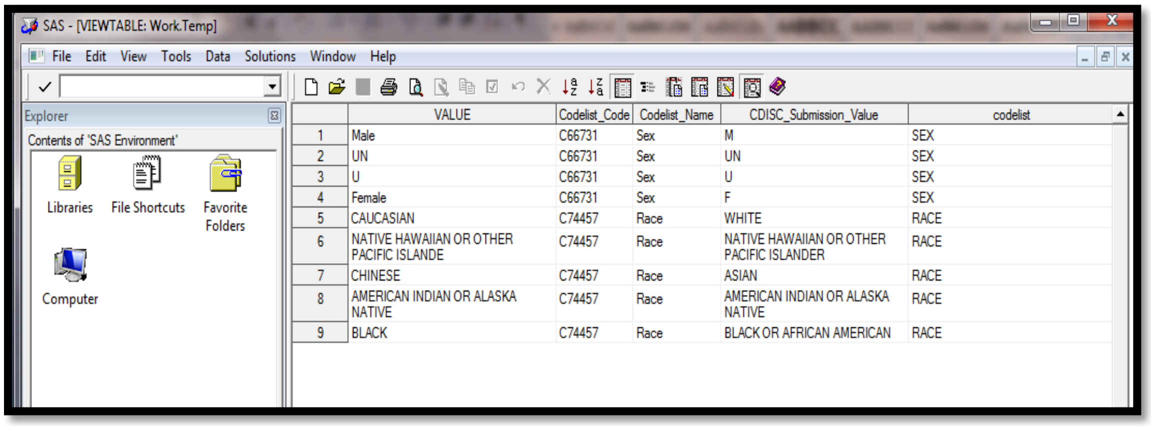
For a demographic data, for example, if the race did not match the CDISC controlled terminology, your data could look something like this:

RAW_RACE	**CDISC SUBMISSION VALUE**
BLACK	BLACK OR AFRICAN AMERICAN
CAUCASIAN	WHITE
CHINESE	ASIAN

You can then create a format assigning the expected CDISC controlled terms:

```
proc format;
       value $race
       'BLACK'     ='BLACK OR AFRICAN AMERICAN'
       'CAUCASIAN' ='WHITE'
       'CHINESE'   ='ASIAN'
       ;
run;
```

Instead of repeating this step for every variable, you can simply create a format dataset by adding a column in the dataset you already have. Below VALUE show the database code list for race and sex as collected in your data:


SAS - [VIEWTABLE: Work.Temp]
File Edit View Tools Data Solutions Window Help
Explorer
Contents of 'SAS Environment'
Libraries
File Shortcuts
Favorite Folders
Computer

	VALUE	Codelist_Code	Codelist_Name	CDISC_Submission_Value	codelist
1	Male	C66731	Sex	M	SEX
2	UN	C66731	Sex	UN	SEX
3	U	C66731	Sex	U	SEX
4	Female	C66731	Sex	F	SEX
5	CAUCASIAN	C74457	Race	WHITE	RACE
6	NATIVE HAWAIIAN OR OTHER PACIFIC ISLANDE	C74457	Race	NATIVE HAWAIIAN OR OTHER PACIFIC ISLANDER	RACE
7	CHINESE	C74457	Race	ASIAN	RACE
8	AMERICAN INDIAN OR ALASKA NATIVE	C74457	Race	AMERICAN INDIAN OR ALASKA NATIVE	RACE
9	BLACK	C74457	Race	BLACK OR AFRICAN AMERICAN	RACE

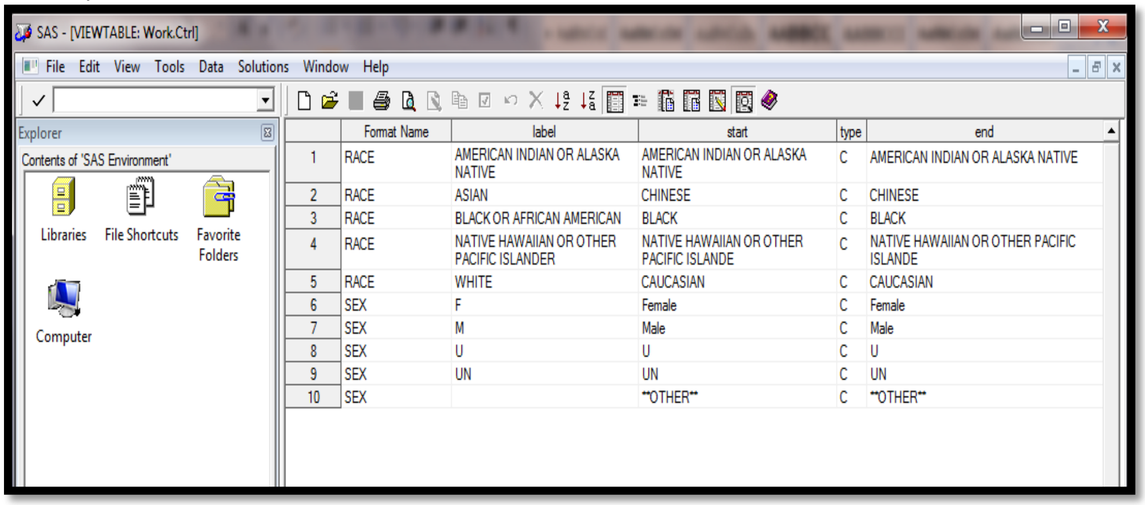
Sample codes for creating a format from a dataset is available on the SAS support website http://support.sas.com

Below is my own version for the dataset I have:

```
proc sql noprint;
create table fmttable as
select unique strip(CODELIST) as fmtname label ='Format Name',
strip(CDISC_SUBMISSION_VALUE) as label,
strip(VALUE) as start,
       from <datasetname>
            ;
quit;

data ctrl;
       set ctfmt end=last;
       end =start;
       output;
       if last then do;
       start='**OTHER**';
       end =start;
```

```
        label=' ';
        output;
        end;
run;

proc format library=work cntlin=ctrl;
run;

quit;
```

The output looks like this:


SAS - [VIEWTABLE: Work.Ctrl]
File Edit View Tools Data Solutions Window Help
Explorer
Contents of 'SAS Environment'
Libraries
File Shortcuts
Favorite Folders
Computer

	Format Name	label	start	type	end
1	RACE	AMERICAN INDIAN OR ALASKA NATIVE	AMERICAN INDIAN OR ALASKA NATIVE	C	AMERICAN INDIAN OR ALASKA NATIVE
2	RACE	ASIAN	CHINESE	C	CHINESE
3	RACE	BLACK OR AFRICAN AMERICAN	BLACK	C	BLACK
4	RACE	NATIVE HAWAIIAN OR OTHER PACIFIC ISLANDER	NATIVE HAWAIIAN OR OTHER PACIFIC ISLANDE	C	NATIVE HAWAIIAN OR OTHER PACIFIC ISLANDE
5	RACE	WHITE	CAUCASIAN	C	CAUCASIAN
6	SEX	F	Female	C	Female
7	SEX	M	Male	C	Male
8	SEX	U	U	C	U
9	SEX	UN	UN	C	UN
10	SEX		**OTHER**	C	**OTHER**

The format can then be applied in any data step:

```
data dm;
 attrib race length=$40;
 set dmraw;
 if not missing(raceval) then race =put(raceval,$race.);
run;
```

<u>QC OF SDTM VARIABLES - CODE:</u>

However, if the data collected should match the controlled terminology (CT) because your data management (DM) team uses the Clinical Data Acquisition Standard Harmonization (CDASH), then you only have to check your SDTM variables against the CDISC controlled terminology.

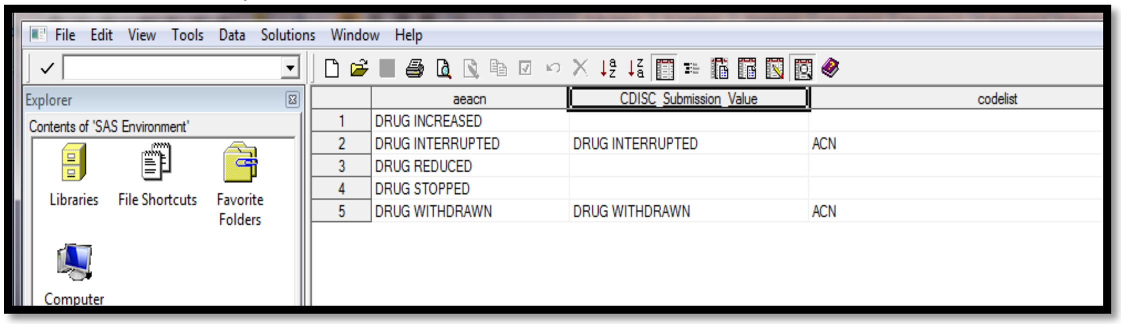
Let us use the Adverse Event (AE) domain as an example.

For example, you can check that AEACN complies with the CDISC CT by following the steps below:

- Select all possible values of AE.AEACN in the created AE domain then merge with the values in the appropriate controlled terminology, ACN.

```
proc freq data =AE noprint;
        Table AEACN/ missing out=AE_ACTION;
run;

proc sql noprint;
        create table AECT as
        select aeacn, b.cdisc_submission_value, b.codelist
        from AE_ACTION left join CTDATA(where=(codelist in ('ACN'))) as b
        on aeacn=b.cdisc_submission_value
        ;
quit;
```

If you have a one-to-one match then your AE domain has the right values in AEACN, otherwise the output will show differences. See example below:



	aeacn	CDISC_Submission_Value	codelist
1	DRUG INCREASED		
2	DRUG INTERRUPTED	DRUG INTERRUPTED	ACN
3	DRUG REDUCED		
4	DRUG STOPPED		
5	DRUG WITHDRAWN	DRUG WITHDRAWN	ACN

HOW TO QUICKLY CHECK DIFFERENCES IN NUMBER OF RECORDS FOR FINDINGS DOMAINS

One issue often encounters when validating data using parallel programming is a difference in number of records. For findings domain, a quick way to identify where the difference occurred is by narrowing it down to which test category and code differ.

Below is an example for the LB domain:

```
proc sql noprint;
        select lbcat,lbscat,lbtestcd,lbtest, count(usubjid) as count
        from sourcelb
        group by lbcat, lbscat, lbtestcd, lbtest
        order by lbcat, lbscat, lbtestcd, lbtest
        ;
quit;
```

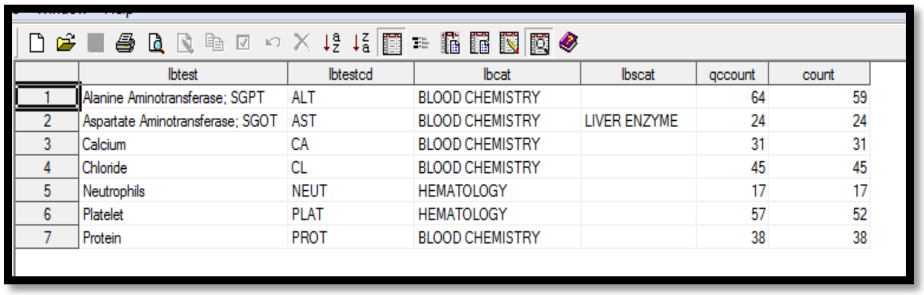
The same result can be achieved with proc freq:

```
proc freq data =sourcelb noprint;
        table lbcat*lbscat*lbtestcd*lbtest / out=qc(keep =lbcat lbscat lbtestcd
lbtest count rename=(count=n)) list missing nocum nopercent;
run;
```

Create a similar dataset for the validation dataset:

```
proc sql noprint;
        create table qs as
        select lbcat,lbscat,lbtestcd,lbtest, count(usubjid) as qccount
        from validlb
        group by lbcat, lbscat, lbtestcd, lbtest
        order by lbcat, lbscat, lbtestcd, lbtest
        ;
quit;

data miss;
        merge qc(in=qc) source(in=dv);
        by lbcat lbscat lbtestcd lbtest;
run;
```

Show dataset here:



	lbtest	lbtestcd	lbcat	lbscat	qccount	count
1	Alanine Aminotransferase; SGPT	ALT	BLOOD CHEMISTRY		64	59
2	Aspartate Aminotransferase; SGOT	AST	BLOOD CHEMISTRY	LIVER ENZYME	24	24
3	Calcium	CA	BLOOD CHEMISTRY		31	31
4	Chloride	CL	BLOOD CHEMISTRY		45	45
5	Neutrophils	NEUT	HEMATOLOGY		17	17
6	Platelet	PLAT	HEMATOLOGY		57	52
7	Protein	PROT	BLOOD CHEMISTRY		38	38

The records where the variables COUNT and QCCOUNT are not equal help you narrow down where the difference comes from.

You can add as many variable levels as needed based on your validation requirements.

If stopping at the test level is not enough, add the visit values for example.

The idea is to pin-point where the difference is as opposed to looking for “a needle in a haystack” since datasets, especially laboratory data, can get very large.

CONCLUSION:

This paper has paper provided examples of SAS code that can be used to automate certain validation programming tasks when creating SDTM datasets.

It can also be used as a starting point for programmers, with no automation tools in place, on how to minimize the amount of repeat programming that often comes with creation of SDTM domains.

As with any new process, starting up is often the most cumbersome. Creating a library of domain attributes and labels as well as one of all the controlled terminologies, CDISC and in-house (sponsor defined), would be most useful; and is almost a necessity if the validation task is to become more efficient.

OpenCDISC will check your SDTM domain, and is an excellent tool for validation but automating programming tasks will help reduce the amount of time checking the report for warning about attributes for example.

REFERENCES:

SAS support website: http://support.sas.com

CDISC: http://www.cdisc.org

OpenCDISC: http://www.opencdisc.org

National Cancer Institute Website: http://www.cancer.gov/cancertopics/cancerlibrary/terminologyresources/cdisc

ACKNOWLEDGMENTS:

This is the text for the acknowledgments. This is the paper body. This is the paper body. This is the paper body. This is the paper body. This is the paper body.

RECOMMENDED READING:

- SDTM Implementation Guide: version 3.1.3 and version 3.1.2
- Study Data Tabulation Model v1.3
- How to use SDTMIG 3.1.3

CONTACT INFORMATION:

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

Name: Dany Guerendo

Enterprise: STATProg LLC
Address: 512 Abbey Fields Loop
City, State ZIP: Morrisville, NC, 27560
Work Phone: 919-423-3560
E-mail: dany.guerendo@statprogllc.com