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:

<table>
<thead>
<tr>
<th>Version</th>
<th>Dlabel</th>
<th>Domain</th>
<th>Vname</th>
<th>Vlabel</th>
<th>Vtype</th>
<th>Core</th>
<th>Vorder</th>
<th>Indomain</th>
</tr>
</thead>
<tbody>
<tr>
<td>v3.1.3</td>
<td>Demographics</td>
<td>DM</td>
<td>STUDYID</td>
<td>Study Identifier</td>
<td>Char</td>
<td>Req</td>
<td>1</td>
<td>D</td>
</tr>
<tr>
<td>v3.1.3</td>
<td>Demographics</td>
<td>DM</td>
<td>DOMAIN</td>
<td>Domain Abbreviation</td>
<td>Char</td>
<td>Req</td>
<td>2</td>
<td>D</td>
</tr>
<tr>
<td>v3.1.3</td>
<td>Demographics</td>
<td>DM</td>
<td>USUBJID</td>
<td>Unique Subject Identifier</td>
<td>Char</td>
<td>Req</td>
<td>3</td>
<td>D</td>
</tr>
<tr>
<td>v3.1.3</td>
<td>Demographics</td>
<td>DM</td>
<td>SUBJID</td>
<td>Subject Identifier for the Study</td>
<td>Char</td>
<td>Req</td>
<td>4</td>
<td>D</td>
</tr>
<tr>
<td>v3.1.3</td>
<td>Demographics</td>
<td>DM</td>
<td>RFSTDTC</td>
<td>Subject Reference Start Date/Time</td>
<td>Char</td>
<td>Exp</td>
<td>5</td>
<td>D</td>
</tr>
<tr>
<td>v3.1.3</td>
<td>Demographics</td>
<td>DM</td>
<td>RFENDTC</td>
<td>Subject Reference End Date/Time</td>
<td>Char</td>
<td>Exp</td>
<td>6</td>
<td>D</td>
</tr>
<tr>
<td>v3.1.3</td>
<td>Demographics</td>
<td>DM</td>
<td>RXFSTDTC</td>
<td>Date/Time of First Study Treatment</td>
<td>Char</td>
<td>Exp</td>
<td>7</td>
<td>D</td>
</tr>
<tr>
<td>v3.1.3</td>
<td>Demographics</td>
<td>DM</td>
<td>RXFSTDTC</td>
<td>Date/Time of Last Study Treatment</td>
<td>Char</td>
<td>Exp</td>
<td>8</td>
<td>D</td>
</tr>
<tr>
<td>v3.1.3</td>
<td>Demographics</td>
<td>DM</td>
<td>RFICDTC</td>
<td>Date/Time of Informed Consent</td>
<td>Char</td>
<td>Exp</td>
<td>9</td>
<td>D</td>
</tr>
</tbody>
</table>
Programming Validation Tips

Prior to using OpenCDISC validator, continued

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

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 it 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.

```sas
%macro sdtm();
filename sdtmct "xxxxxxx/cdisc_v_312.csv";

*-----------------------------------------------*
* Read in excel spreadsheet
* Output permanent dataset with all domains attributes
*-----------------------------------------------*;
proc import out= sdtmct 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.
Programming Validation Tips

Prior to using OpenCDISC validator, continued

**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.

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

data attrbut(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 =attrbut;
  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
```

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.

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

data attrbut(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 =attrbut;
  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;
Programming Validation Tips Prior to using OpenCDISC validator, continued

*---------------------------------------------------------------------*
* 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 =, dsort =);

An example of a demographic domain created using the macro ATTRIBUT.
**QUALITY CHECK ON CONTROLLED TERMINOLOGY**

One of the most common issue we encounter is, when controlled terminology is applied, how do we recognize 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.

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

```sas
filename sdtmct "xxxxxxxxxxxxx/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:

<table>
<thead>
<tr>
<th>C17998</th>
<th>C66767</th>
<th>ACN</th>
<th>Action Taken with Study Treatment</th>
<th>UNKNOWN</th>
<th>U; Unknown</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>No</td>
<td>OUT</td>
<td>OUT</td>
<td>OUT</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Outcome of Event</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>RAW_RACE</th>
<th>CDISC_SUBMISSION_VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>BLACK</td>
<td>BLACK OR AFRICAN AMERICAN</td>
</tr>
<tr>
<td>CAUCASIAN</td>
<td>WHITE</td>
</tr>
<tr>
<td>CHINESE</td>
<td>ASIAN</td>
</tr>
</tbody>
</table>

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

```sas
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:

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

Below is my own version for the dataset I have:

```sas
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;
run;
```
Programming Validation Tips

Prior to using OpenCDISC validator, continued

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

proc format library=work cntlin=ctrl;
run;
quit;
```

The output looks like this:

![Format Table]

The format can then be applied in any data step:

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

**QC OF SDTM VARIABLES - CODE:**

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.

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

```sas
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:

![SAS Output](image)

**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:

```sas
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:

```sas
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:

```sas
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;
```

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

Prior to using OpenCDISC validator, continued

Show dataset here:

<table>
<thead>
<tr>
<th>best</th>
<th>bested</th>
<th>best</th>
<th>best</th>
<th>account</th>
<th>count</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Antriminurese</td>
<td>ALT</td>
<td>BLOOD CHEMISTRY</td>
<td>64</td>
<td>59</td>
</tr>
<tr>
<td>2</td>
<td>Apetitive Antriminurese</td>
<td>AST</td>
<td>BLOOD CHEMISTRY</td>
<td>24</td>
<td>24</td>
</tr>
<tr>
<td>3</td>
<td>Cenium</td>
<td>CA</td>
<td>BLOOD CHEMISTRY</td>
<td>31</td>
<td>31</td>
</tr>
<tr>
<td>4</td>
<td>CLEAN</td>
<td>CL</td>
<td>BLOOD CHEMISTRY</td>
<td>45</td>
<td>45</td>
</tr>
<tr>
<td>5</td>
<td>Nitrophili</td>
<td>NEUT</td>
<td>HEMATOLOGY</td>
<td>17</td>
<td>17</td>
</tr>
<tr>
<td>6</td>
<td>Paleed</td>
<td>FLAT</td>
<td>HEMATOLOGY</td>
<td>57</td>
<td>52</td>
</tr>
<tr>
<td>7</td>
<td>Poison</td>
<td>PROT</td>
<td>BLOOD CHEMISTRY</td>
<td>38</td>
<td>38</td>
</tr>
</tbody>
</table>

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