Creating SDTMs and ADaMs CodeList Lookup Tables

Sunil Gupta, TalentMine, Raleigh, NC

ABSTRACT

Do you need to review and confirm codelist values for variables in SDTMs and ADaMs? Codelists are lists of unique values for key variables such as LBTEST, AVISIT and AVISITN. Codelists need to be cross-referenced with control terms.

Codelist dictionary compliance checks are very important but are often neglected. Since all raw data is now standardized to control terms, there are many opportunities for cross checking SDTM, ADaMs with codelist dictionary tables. In addition, the define xml file must have a correct and updated codelist section.

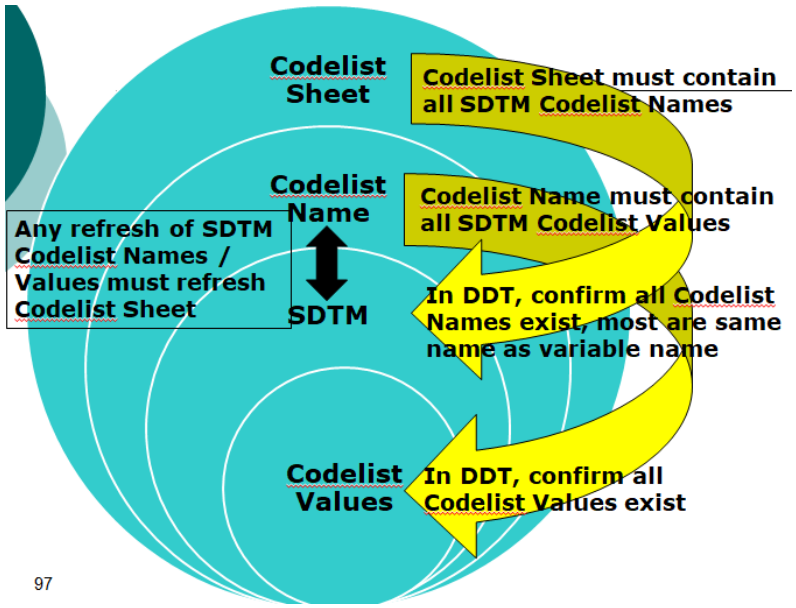
INTRODUCTION

Codelist dictionaries are the heart of all SDTMs and ADaMs. While there is focus to convert raw data to control terminology, there are minimum efforts to review and document codelist dictionaries. Most methods involve a manual process which is time consuming and not error proof. In addition, it is more effective to review codelist dictionaries in a central file to better identify inconsistencies.

This paper shows how to automatically create a codelist dictionary across all ADaMs and SDTMs as well as compare codelist dictionaries from SDTMs and define xml specifications for example. Both examples are essential for meeting CDISC compliance. Without an automated process to create codelist dictionaries, the alternative method of applying Proc FREQ on all categorical variables is very time consuming.

CODELIST DICTIONARIES

Creating codelist dictionaries can be automated by accessing SAS dictionary tables. Codelist dictionaries need to be refreshed since SDTMs and raw data are refreshed on a regular basis. The diagram below shows how codelist values are the core to having consistency between SDTMs across studies as well as SDTM specifications.



The example below creates a dictionary excel file of all ADaM codelists as columns. Since ADaMs contain both continuous and categorical variables, additional code is included to exclude unwanted ADaMs or continuous variables for which codelists do not make sense, else the default is to include all ADaMs and variables. This is useful to see all unique values across all ADaMs. This allows cross referencing ADaM codelist with the control terminology.

The code below can also be applied to create SDTM codelist dictionaries. Note that this example contains advanced macro programming to create and loop through records dynamically creating macro calls for each ADaM and variable. Each intermediate dataset is then merged together in the same correct order so that codelists are grouped

by ADaMs and variable order. Once all of the macro calls are created in a separate file, the file is included in the program and executed.

```
* Automatically create dataset codelist excel file;
libname dsn '\\analysis\data\adam' access=readonly;

* for codelist differences make macro call and copy codelist from excel file;
%MACRO CODEV(libn, dsn, varn, runnum=2)/minoperator;

%if &varn=LBCAT or &varn=LBNAM or &varn=LBSCAT or &varn=LBSTAT or &varn=LBTEST or &varn=LBTESTCD
%then %do;
ods output OneWayFreqs=&dsn._&varn(rename =(&varn = &dsn._&varn) keep = &varn);
proc freq data=&libn..&dsn;
  tables &varn/nocol norow nocum nopercent;
run;
%end;
%else %do;
ods output OneWayFreqs=&varn(keep = &varn);
proc freq data=&libn..&dsn;
  tables &varn/nocol norow nocum nopercent;
run;
%end;

* account for same variables across domains;
%if &runnum=1 %then %do;
data codev;
set &varn;
run;
%end;
%else %if &runnum=2 and &varn &varn^=LBCAT and &varn^=LBNAM and &varn^=LBSCAT and &varn^=LBSTAT
and &varn^=LBTEST and &varn^=LBTESTCD  %then %do;
data codev;
merge codev &varn;
run;
%end;
%else %if &runnum=2 and &varn=LBCAT or &varn=LBNAM or &varn=LBSCAT or &varn=LBSTAT or &varn=LBTEST
or &varn=LBTESTCD  %then %do;
data codev;
merge codev &dsn._&varn;
run;
%end;
%mend codev;
%*CODEV(dsn, adsl, siteid , runnum=1);
%*CODEV(dsn, adsl, aphase );

* create metadata of dataset and variable names;
* delete all dates, num vars, etc;
* and memname in ('DM' 'AE' 'LBAL');
* create lookup table to exclude metadata;
*MBORRES MBSTRESC MBSTRESN MHDECOD MHENRTPT MHENTPT MHHLGT MHHLGTCD MHHLT
MHHLTCD MHLLT MHTERM
PCORRES    PCORRESU PCREFID PCSTRESC PCSTRESN VSORRES VSSTRESC VSSTRESN;

proc sql;
  create table clxmcall as
    select unique memname as dsn, name as var
    from sashelp.vcolumn where upcase(libname) = 'DSN' and memname in ('ADPCR' 'ADPRCS' 'ADBMK' 'ADBMKS'
'ADCSF' 'ADCSFF')
     and name ^in ('USUBJID' 'SUBJID' 'AETERM' 'LBORNRHI' 'LBORNRLO' 'LBORRES' 'LBSTNRHI' 'LBSTNRLO'
'LBSTRESC' 'LBSTRESN' 'AEHLGT' 'AEHLGTCD' 'AEHLT' 'AEHLTCD' 'AELLT' 'AELLTCD' 'AEPTCD' 'AEDECOD'
'TRORRES' 'TRSTRESC' 'TRSTRESN' 'CMTRT' 'DAORRES' 'DASTRESC' 'DASTRESN' 'ADT' 'AVAL' 'CHG' 'BASE'
'AUCAVAL'   'DAY0VAL' 'PEAKVAL' 'TTPEAK' 'FCHG')
    and index(name, 'DT') = 0 and index(name, 'TM') = 0 and index(name, 'SEQ') = 0 and index(name, 'DY') = 0
    order by memname, name;
quit;
```

```
* write to macro calling file;
data _null_;
set clxmcall;

file "\\analysis\dev\program\utility\codelist_ds_bmk.sas";

if _n_ = 1 then
put '%codev(dsn,'
dsn
', '
 var
', runnum=1)';
else
put '%codev(dsn,'
dsn
', '
 var
')';
run;

* metadata macro calls;
%inc "\\analysis\dev\program\utility\codelist_ds_bmk.sas";

ods html file="\\analysis\dev\program\utility\codelist_ds_bmk_&sysdate9..xls";
proc print data=codev noobs;
var _all_;
run;
ods html close;
```

This SAS generated output shows the results of the generated macro calls for each dataset and variable combination. Once this SAS program is created, it is executed. This is an essential part to automate the process so the program is robust and dynamic for any number of datasets. In the first macro call, RUNNUM is set to 1 since this is the first dataset.

SAS Generated Output

```
%codev(dsn,ADBMK , ABLFL , runnum=1)
%codev(dsn,ADBMK , ALSFL )
%codev(dsn,ADBMK , APERIOD )
%codev(dsn,ADBMK , APHASE )
%codev(dsn,ADBMK , APHASEN )
%codev(dsn,ADBMK , ARM )
%codev(dsn,ADBMK , ASSAYMET )
%codev(dsn,ADBMK , ASSAYNO )
%codev(dsn,ADBMK , AVISIT )
%codev(dsn,ADBMK , AVISITN )
%codev(dsn,ADBMK , BACKBS )
%codev(dsn,ADBMK , CCFL )
%codev(dsn,ADBMK , CMMT )
```

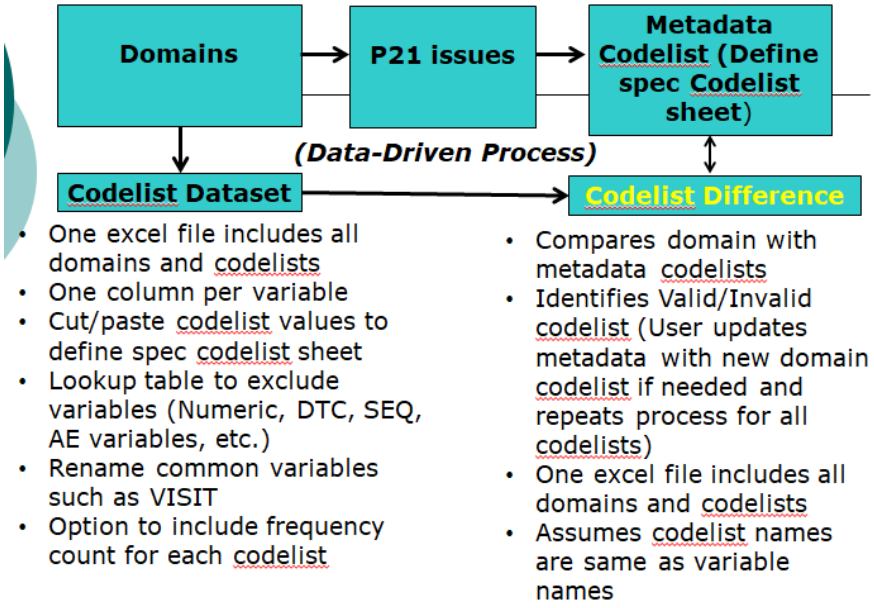
This output shows the excel file with columns for each variable. The values listed in each row are in alphabetical sort order of all unique values. The column order is same order as the macro calls in the SAS program. The columns represent the codelist dictionary for selected datasets and variables. In this excel file, it is easy to review, search and confirm codelists.

Output

	A	B	C	D	E	F	G	H	I	J
1	ABLFL	ALSFL	APERIOD	APHASE	APHASEN	ARM	ASSAYMET	ASSAYNO	AVISIT	AVISITN
	Y	Y	1	Phase1	1	KTE-C19	MILLIPLEX MAP Human CD8+ T Cell Magnetic Bead Panel custom kit, PN HCD8MAG-15K	DR2A00	BASELINE	0
2			2	Phase2 Cohort1	2.1		MSD V-PLEX Plus Human Biomarker 40 Plex Kit, PN K15209G-4	DRA00B	DAY 7	7
3			.	Phase2 Cohort2	2.2		R&D Systems Human CD25/IL-2Ra Quantikine ELISA, PN DR2A00	ELLA-2	MONTH 12	14
4			.	.			R&D Systems Human IL-1Ra/IL-1F3 Quantikine ELISA, PN DRA00B	ELLA-F	MONTH 15	28
5			.	.			Simple Plex™ Human Ferritin Cartridge	HCD8MAG-15K	MONTH 18	90
6			.	.				K15209G-4	MONTH 24	180
7			.	.				K15209G-5	MONTH 3	270
8			.	.				ab108837	MONTH 6	360
9			.	.					MONTH 9	450
10			.	.					WEEK 2	540
11			.	.					WEEK 4	720
12										

COMPARING CODELIST DICTIONARIES

Comparing codelist dictionaries can be automated by accessing SAS dictionary tables and applying macro programming techniques to effectively compare codelists between two datasets. SDTM codelist dictionaries need to be compared to SDTM specs and the control terminology dictionary to identify any inconsistencies. The diagram below shows how comparing codelists is a data-drive process since one source is based on clinical data and another source is a reference to a unique listing of values.



The example below compares differences in codelists. Codelists from ADaMs need to be a subset of the ADaM define specifications since the codelist in define.xml file may display extra codes that are within the codelist but not in ADaMs. This example contains advanced macro programming to create and loop through records dynamically creating macro calls for each ADaM and variable. Each intermediate dataset is then merged together in the same order as called so that codelists are grouped by ADaMs and variable order. Once all of the macro calls are created in a separate file, the file is included in the program to be executed. Proc IMPORT converts the codelist sheet in define spec to CODELIST1.

```
* Automatically compare dataset with define spec codelist;
libname dsn '\\analysis\data\adam' access=readonly;

options mprint mlogic symbolgen;

* read define spec;
* adams read id term;
```

```
proc import datafile="\\analysis\data\adam\xpt\define\define.xlsx"
            out=codelist1 (keep =  id term) dbms=xlsx replace;
             sheet="codelists";
             getnames=yes;
run;

* adams;
data codelist (rename=id2=id);
length name $10. id2 $10.;
set codelist1;

name = scan(id, 1);
id2 = scan(id, 2);
drop id;
where id > '';
run;

proc sort data=codelist nodupkey out=clmcall1;
by name;
where name =: 'AD';
run;

data clmcall;
set clmcall1;

* for adams;
dsn=name;

* sdtms;
*dsn=id;
*dsn=substr(id, 1, 2);
* include only matching domain names in codelist;
*where id in: ('AE' 'CM' 'CO' 'DA' 'DD' 'DM' 'DS' 'DV' 'EG' 'EX' 'FA'  'HO' 'IE' 'IS' 'LB' 'MB' 'MH'
            'MI' 'MO' 'PC' 'PE' 'PR' 'QS' 'RS' 'SE' 'SS' 'SV' 'TA' 'TE' 'TI' 'TR' 'TS' 'TU' 'TV' 'VS' 'XC');
*keep id dsn;

* note that LB is split into three versions LBAL, LBCY and LBLY;
if dsn='LB' then do;
dsn='LBAL'; output;
dsn='LBCY'; output;
dsn='LBLY'; output;
end;
else output;
run;

%macro ct_list(dsn=adsl, varn=race);
proc sql noprint;
* control term dataset for metadata checking;
select unique term into :mt_ctlist separated by ', ' from codelist where id=upcase("&varn");

* excludes missing values;
select unique &varn into :ds_ctlist separated by ', ' from dsn.&dsn where &varn > '';

create table dc as
select unique  "&dsn" as domain,
  compbl("&varn Codelist") as key_grp_vr length=50
  , "&ds_ctlist" as ds_ct label="Domain Codelist:" length=300
  , "&mt_ctlist" as mt_ct label="Control Terms Codelist:" length=300
  , case when "&ds_ctlist" ^= "&mt_ctlist" then 'Invalid Codelist'
  else 'Valid Codelist' end as ctck label="&varn Codelist"
  from dsn.&dsn;
quit;

proc append base= dc_all data=dc force;
run;
```

```
%mend ct_list;

* write to macro calling file;
data _null_;
set clmcall;

file "\\analysis\dev\program\utility\codelist_dif.sas";

put '%ct_list(dsn='
dsn
', varn='
 id
')';
run;

* required to create dummy dataset for proc append;
data dc_all;
  length domain $20. key_grp_vr $50. ds_ct $500. mt_ct $500. ctck $50. define_update $20.;
run;

* metadata macro calls;
%inc "\\analysis\dev\program\utility\codelist_dif.sas";

proc sort data= dc_all nodupkey out=dc_all1;
by _all_;
run;

data dc_all1;
set dc_all1;
if ctck = 'Valid Codelist' then define_update = 'n/a';
where domain > '';;
run;

ods html file="\\analysis\dev\program\utility\codelist_dif_&sysdate9..xls";
proc print data=dc_all1 noobs;
var _all_;
run;
ods html close;
proc print; run;
```

Similar to the previous example, the SAS generated output shows the results of the generated macro calls for each dataset and variable combination. Once this SAS program is created, it is executed. This is an essential part to automate the process so the program is robust and dynamic for any number of datasets.

SAS Generated Output

```
%ct_list(dsn=ADAE , varn=KREL30FL )
%ct_list(dsn=ADDA , varn=PARAM )
%ct_list(dsn=ADEFF , varn=PARAM )
%ct_list(dsn=ADEX , varn=PARAM )
%ct_list(dsn=ADIS , varn=PARAM )
%ct_list(dsn=ADLB , varn=PARAM )
%ct_list(dsn=ADMB , varn=PARAM )
%ct_list(dsn=ADPCR , varn=PARAM )
%ct_list(dsn=ADPCRS , varn=PARAM )
%ct_list(dsn=ADQS , varn=PARAM )
%ct_list(dsn=ADRS , varn=PARAM )
%ct_list(dsn=ADSAF , varn=PARAM )
%ct_list(dsn=ADSL , varn=SEX )
%ct_list(dsn=ADTR , varn=PARAM )
%ct_list(dsn=ADTTE , varn=PARAM )
```

In addition, this output shows the excel file with columns for each variable. The values listed in each row are in alphabetical sort order of all unique values. The column order is same order as the macro calls in the SAS program. The columns represent the codelist dictionary for selected datasets and variables. In this excel file, it is easy to review, search and confirm codelists. CTCK displays VALID or INVALID CODELIST for each ADaM and variable. For INVALID CODELIST rows, lists DS_CAT from ADaM and MT_CAT from define specification can be compared and updated.

Output

	A	B	C	D	E	F
1	**domain**	**key_grp_vr**	**ds_ct**	**mt_ct**	**ctck**	**define_update**
2	ADAE	KREL30FL Codelist	N, Y	N, Y	Valid Codelist	n/a
3	ADDA	PARAM Codelist	CD3 Cells, CD4 Cells, CD4/CD8 Ratio, CD8 Cells, Central Memory Cells, Dispensed Amount, Effector Cells, Effector Memory Cells, Infused Amount, Interferon Gamma, KTE-C19 Delivery, KTE-C19 Time from Leukapheresis to Delivery at Site, Manufacturing Dose Produced, Naive Cells, Patient Weight at IP admin	% B Cells of Viable Leukocytes (%), Absolute Basophils Count (10^9/L), Absolute Eosinophils Count (10^9/L), Absolute Monocyte Count (10^9/L), Alanine Aminotransferase (U/L), Albumin (g/L), Alkaline Phosphatase (U/L), Antibody Test (pg/mL), Aspartate Aminotransferase (U/L), Basophils/Leukocytes (%),	Invalid Codelist	
4	ADEFF	PARAM Codelist	Best Overall Resp. by Cent. Read Per. 01, Best Overall Resp. by Cent. Read Per. 02, Best Overall Response by Inv. Per. 01, Best Overall Response by Inv. Per. 02, First Obj. Response in Per. 01 by Central Read, First Obj. Response in Per. 01 by Investigator, First Obj. Response in Per. 02 by Central	% B Cells of Viable Leukocytes (%), Absolute Basophils Count (10^9/L), Absolute Eosinophils Count (10^9/L), Absolute Monocyte Count (10^9/L), Alanine Aminotransferase (U/L), Albumin (g/L), Alkaline Phosphatase (U/L), Antibody Test (pg/mL), Aspartate Aminotransferase (U/L), Basophils/Leukocytes (%),	Invalid Codelist	

SUMMARY

The automatic creation and comparison of codelist dictionaries helps organizations to better identify inconsistencies as well as enforce CDISC compliance. The examples presented work for both SDTM and ADaMs. The standard framework can be customized for more tailed and targeted review.

REFERENCES

Gupta, Sunil, “Ready To Become Really Productive Using PROC SQL?”, PharmaSUG 2012, SAS Global Forum 2011 and WUSS 2010

CONTACT INFORMATION

The author welcomes your comments and suggestions.
Sunil Gupta
E-mail: Sunil.Gupta@TalentMine.net

Sunil Gupta, MS, is an international speaker, best-selling SAS author, and a global CDISC corporate trainer. Sunil is Executive Director of Data Sciences at TalentMine and has over twenty-five years of experience in the pharmaceutical industry. Most recently, Sunil is a CDISC Oncology ADaM reviewer and has taught his CDISC online class at the University of California at San Diego and SAS Institute India. Sunil also taught his Sharpening Your SAS Skills online class for UCLA Extension. In 2011, Sunil launched his unique SAS resource blog, SASSavvy.com, for smarter SAS searches and teaches Data Science Using SAS. Sunil has MS in Bioengineering from Clemson University and a BS in Applied Mathematics from the College of Charleston.