PharmaSUG 2014 – Paper CC14

A Toolkit to Check Dictionary Terms in SDTM

Huei-Ling Chen, Tech Data Services, Bridgewater, NJ
Helen Wang, Sanofi, Bridgewater, NJ

ABSTRACT

WHO Drug Dictionary (WHO-DD) and MedDRA are two important dictionaries used in the SDTM datasets. In the intervention domains, WHO-DD is used to derive the standardized medication name. In the event domains, the standardized text description of an event is based on dictionary MedDRA. A standardized dictionary term can be derived from an original term when its value exists in the dictionaries. Sometimes an original term fails to facilitate coding due to its value does not exist in the dictionary. In a busy project team, preparing SDTM packages for multiple studies is very common. It will be helpful if there is a checking toolkit to quickly identify records that fail to have dictionary term derived when preparing SDTM packages. This paper will first briefly summarize the application of WHO-DD and MedDRA and list out the dictionary-derived variables commonly used in clinical trial studies. Then a method to check every SDTM domains with dictionary WHO-DD or MedDRA derived variables is described here. The method is written into a macro format which can check the entire SDTM package without pre-requisite knowledge on which domain has dictionary derived terms. Another benefit of this handy toolkit is that it is a portable macro that can easily be adapted and adopted in other applications.

KEYWORDS

SDTM, CDISC, MEDDRA, WHO DD, OPEN CDISC, SASHELP

INTRODUCTION

The Center for Drug Evaluation and Research (CDER) of FDA has encouraged the sponsors (e.g. pharmaceutical companies) to use standard data format when submitting a new drug application. Collaborating with CDER, CDISC has developed SDTM dataset structure as a guideline for the sponsors to follow. Two coding dictionaries have been used broadly in the SDTM datasets. WHO-DD is for the interventions domains. MedDRA is for the events domains. Discussions involve preferred term derived by the sponsor from the coding dictionary. Interventions domains include but not limited to the following domains – CM, SU, etc. Events domains include AE, MH, CE, etc. Some dictionary terms are required variables in SDTM. When such required dictionary terms are not derived due to the value of original terms do not exist in the dictionary, OPENCDISC tool would trigger violation messages in the review log. One handy macro is introduced here as a toolkit to help programmers review the SDTM domains specifically for the dictionary-derived variables. It can save the troubleshooting time and effort on the OPENCDISC tool to examine the data. It lists out records with required decode values not properly derived.

TOOLS: DICTIONARY TABLES AND SASHELP VIEWS

This paper uses the SASHELP.VCOLUMN to retrieve the SDTM datasets information from the assigned data library.

SAS user can use either SAS DICTIONARY tables or SASHELP views to quickly get information on the datasets, variables, and formats from the current SAS session. The DICTIONARY tables can be accessed by SQL language only. However, SAS provides PROC SQL views based on the DICTIONARY tables and store these views in the SASHELP library. These views can be accessed by DATA step or SQL. SASHELP.VCOLUMN contains 18 variables. Commonly used variables are LIBNAME, MEMNAME, MEMTYPE, NAME, TYPE, LENGTH, etc. Below is an example of a SASHELP.VCOLUMN for SDTM dataset AE.

libname	memname	memtype	name	type	length
ADSD	AE	DATA	STUDYID	char	15
ADSD	AE	DATA	DOMAIN	char	2
ADSD	AE	DATA	USUBJID	char	25
ADSD	AE	DATA	AESEQ	num	8
ADSD	AE	DATA	AESPID	char	6
ADSD	AE	DATA	AETERM	char	200
ADSD	AE	DATA	AEMODIFY	char	200
ADSD	AE	DATA	AELLT	char	120
ADSD	AE	DATA	AELLTCD	num	8
ADSD	AE	DATA	AEDECOD	char	120

WHO-DD VS. INTERVENTIONS DOMAIN

WHO-DD is a drug dictionary which contains several tables (datasets) to store the drug names, formulations, ingredients, ATC classes. The unique key variable is a combination of three variables - Drug Record Number, SEQ1 (Sequence Number 1), and SEQ2 (Sequence Number 2). Variable SEQ1 identifies the salt or the ester of the active ingredient. Variable SEQ2 identifies the different names for the drug, could be trade name or generic name.

When collecting the interventions information from a patient, the original drug name is entered. The original drug name (e.g. CMTRT) can be a free text. Accompanied with the free text drug name, a Drug Record Number, SEQ1, and SEQ2 should be identified and entered into the data.

Interventions domains include CM, EX, SU, and any other sponsor-defined interventions domains. Merged by the unique combination of Drug Record Number, SEQ1, and SEQ2 to the dictionary, the drug preferred term (--DECOD) and ATC code can be retrieved from the dictionary.

Original Verbatim	Modified Verbatim	Mapped Variables	Dictionary Terms
--TRT (e.g.CMTRT)	--MODIFY (e.g. CMMODIFY)	RECNO SEQ1 SEQ2	--DECOD = Preferred Term ATC Code (e.g. CMCLAS = Medication Class CMCLASCD = Medication Class Code)

MEDDRA VS. EVENTS DOMAIN

Events domains include AE, MH, DV, CE, and any other sponsor-defined events domains.

Variable --DECOD is the preferred term in MedDRA. In SDTM, the variable AEDECOD and MHDECOD, in AE domain and MH domain, respectively, are the required variables.

Other higher grouping variables which often are permissible variables include High Level Group Term, High Level Term.

Often accompanied with the original verbatim terms, variable –LLTCD, Lower Level Term, is also provided in the raw dataset. A coding group that is specialized in manually select the appropriate –LLTCD for the unmapped –TERM. Programming group usually has –LLTCD ready to map with the MedDRA dictionary.

Original Verbatim	Modified Verbatim	Mapped Variables	Dictionary Terms
--TERM (e.g. AETERM MHTERM)	--MODIFY (e.g. AEMODIFY MHMODIFY)	--LLTCD (e.g. AELLTCD MHLLTCD)	--DECOD = Preferred Term --HLGT = High Level Group TERM --HLT= High Level TERM --LLT= Lowest Level TERM --LLTCD = Lowest Level TERM Code --PTCD = Preferred TERM Code --HLTCD = High Level TERM Code --HLGTCD = High Level Group TERM Code --SOCCD = System Organ Class Code

OPENCDISC

OpenCDISC would trigger error/warning messages when it identifies that certain criteria are not meet by the data during the review. For example, issue SD0008 (Value for --DECOD not found in MedDRA dictionary) is created in the OpenCDISC report when a MHDECOD is not found in the MH domain. Below is an example from 'Details' spreadsheet in an OpenCDISC report.

Domain	Record	Count	Variables	Values	Rule ID	Message	Category	Severity
MH		1	MHDECOD	Amygdalohippocampectomy	SD0008	Value for MHDECOD not found in MedDRA	Terminology	Error

						dictionary		

The Rule is described in the 'Rules' spreadsheet.

Rule ID	Message	Description	Category	Severity
SD0008	Value for --DECOD not found in MedDRA dictionary	Value for the Dictionary-Derived Term (--DECOD) variable must be populated using a Preferred Term of the MedDRA dictionary of a version specified in the define.xml (Case-insensitive)	Terminology	Error

When a study size is big, it can be time consuming to run OpenCDISC tool. With a handy macro to check the entire SDTM package without pre-requisite knowledge on which domain has dictionary derived terms, can help save trouble shooting time on the data issues.

KEY SYNTAX FOR MACRO %CHKDERIVED (INLIB=)

Step 1: Use SASHELP.VCOLUMN to retrieve the LIBNAME="&INLIB" and MEMNAME="DATA" record. Restrict the interested variables only to the variables in the dictionary WHO-DD or MedDRA.

```
data meta;
    set sashelp.vcolumn(where=(libname="&inlib" and memtype="DATA" ) );
    if substr(left(name),3) in ('TERM' 'TRT'
                                'DECOD' 'PTCD' 'SOCCD'
                                'HLGT' 'HLGTCD'
                                'HLT' 'HLTCD'
                                'LLT' 'LLTCD'
                                ) ;
run;
```

Step 2: Check datasets only when both raw variable (--TERM or --TRT) and at least one dictionary variable (--DECOD, --PTCD, --SOCCD, --HLGT, --HLGTCD, --HLT, --HLTCD, --LLT, and --LLTCD) are available in the datasets.

```
proc sql noprint;
    create table meta1 as
    select distinct memname, count(memname) as orgvar from meta
    where substr(left(name),3) in ('TERM' 'TRT')
    group by memname
    order by memname;

    create table meta2 as
    select distinct memname, count(memname) as dervar from meta
    where substr(left(name),3) in ('DECOD' 'PTCD' 'SOCCD'
                                'HLGT' 'HLGTCD'
                                'HLT' 'HLTCD'
                                'LLT' 'LLTCD')
    group by memname
    order by memname;

quit;

data both;
    merge meta1 meta2;
    by memname;
    if orgvar>0 and dervar>0;

run;
```

Step 3: Assign each of these check-able dataset names to a corresponding macro parameter.

```
proc sql noprint;
    select count(*) into :tot
    from both
    ;
quit;

%let tot=&tot;

proc sql noprint;
    select memname
    into :dat1 - :dat&tot
    from both
    ;
quit;
```

Step 4: Examine these check-able datasets one by one. For each dataset, assign each of these identified variables with a corresponding macro parameter. Print out records with dictionary variables not derived but with non-blank --TERM or --TRT values.

```
%do i=1 %to &tot ;

    proc sql noprint;
        select name into :orgvar
        from meta
        where memname="&&dat&i" and substr(left(name),3) in
          ('TERM' 'TRT')
        ;
    quit;

    proc sql noprint;
        select count(*) into :vartot
        from meta
        where memname="&&dat&i" and
              substr(left(name),3) in ('DECOD' 'PTCD' 'SOCCD'
                                       'HLGT' 'HLGTCD'
                                       'HLT' 'HLTCD'
                                       'LLT' 'LLTCD')
        ;
    quit;

    %let vartot=&vartot;

    proc sql noprint;
        select name
        into :var1 - :var&vartot
        from meta
        where memname="&&dat&i" and
              substr(left(name),3) in ('DECOD' 'PTCD' 'SOCCD'
                                       'HLGT' 'HLGTCD'
                                       'HLT' 'HLTCD'
                                       'LLT' 'LLTCD')
        ;
    quit;

    %do j=1 %to &vartot;
        data c&j (keep= usubjid &orgvar &&var&j) ;
            set &inlib..&&dat&i;
            if &orgvar ne '' and &&var&j = '';
        run;
```

```
        proc sort data=c&j;
            by usubjid &orgvar;
        run;
    %end;

    data _prt;
        retain usubjid &orgvar;
        merge %do j=1 %to &vartot; c&j %end;;
        by usubjid &orgvar;
    run;

    proc print;
        title "&inlib..&&dat&i has the following uncoded
               --DECOD or high level variables ";
    run;

%end;
```

OUTPUT

Execute this macro by specifying the library name.

```
%chkDerived(inlib = SDTMLIB);
```

The output would list out datasets with records that fail to have dictionary terms derived. For intervention domains, variable USUBJID, --TRT, and –DECOD would be in the list. Below is an example from CM domain.

SDTMLIB.CM has the following uuencoded --
DECOD or high level variables

Obs	USUBJID	CMTRT	CMDECOD
1	111111-999-888-777	EMCONCOR 2.5 MG	
2	111111-999-888-666	VIBTIL	
3	111111-999-888-555	PANTOMED 20 MG	
4	111111-999-888-444	AMUFORT (PELARGONIUM)	
5	111111-999-888-333	BETACOR	

For event domains, variable USUBJID, --TERM, –DECOD, and the higher level group terms would be in the list. Below is an example from MH domain.

SDTMLIB.MH has the following uuencoded --
DECOD or high level variables

Obs	usubjid	MHTERM	MHLLT	MHLLTCD	MHDECOD	MHPTCD	MHHLT
1	11111-888-777-666	NASAL POLYPOSIS		.		.	
2	11111-888-777-666	ASTHMA		.		.	
3	11111-888-666-555	NASAL POLYPOSIS		.		.	
4	11111-888-777-669	SURGERY FOR NASAL POLYPOSIS		.		.	
5	11111-888-777-670	SURGERY FOR NASAL POLYPOSIS		.		.	

CONCLUSION

Many possibilities could cause a failed encoding. Sometimes it was due to an old version of MedDRA / WHO-DD dictionary database was mistakenly used in the data processing stage. Or it could be a treatment drug was not identified with a suitable Drug Record Number. The macro provided here can quickly search the chosen library and automatically identify which dataset is an event or intervention domains with check-able variables. With the output list created by the macro, the eventual solution of a proper coding is to identify the correct [Drug Record Number + SEQ1 + SEQ2] for the drug, or to have the correct LLTCD for the events.

REFERENCES

- Gerlach, J. (2005) "A Better Perspective of SASHELP Views" Proceedings of the PharmaSUG 2005 Conference
- Peppard, T. (2007) "Using the WHO Drug Dictionary for Reporting Clinical Trials," Proceedings of the MWSUG 2007 Conference.
- Lafler, K. (2010) "Exploring DICTIONARY Tables and SASHELP Views," Proceedings of the SAS Global Forum 2010 Conference

ACKNOWLEDGMENTS

The authors would like to thank the B&P management team of Sanofi-Aventis for their advice on this paper/presentation.

CONTACT INFORMATION

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

Huei-Ling Chen
Tech Data Services
55C-330A, 55 Corporate Drive
Bridgewater, NJ 08807
Phone: 908-981-6871
e-mail: huei-ling.chen@sanofi.com

Helen Wang
Sanofi
55C-330A, 55 Corporate Drive
Bridgewater, NJ 08807
Phone: 908-981-4752
e-mail: helen.wang@sanofi.com

TRADEMARK

APPENDIX

Example code

```
%macro chkDerived(inlib = );

%let inlib=%upcase(&inlib);

** retrieve all data/var name from sashelp.vcolumn;
data meta;
    set sashelp.vcolumn(where=(libname="&inlib" and memtype="DATA" ) );
    if substr(left(name),3) in ('TERM' 'TRT' 'DECOD' 'HLGT' 'HLT' 'LLT' 'LLTCD'
                                'PTCD' 'HLTCD' 'HLGTCD' 'SOCCD') ;
run;
```

```
** this macro check only variables are check-able;
proc sql noprint;
    create table meta1 as
    select distinct memname, count(memname) as orgvar from meta
    where substr(left(name),3) in ('TERM' 'TRT')
    group by memname
    order by memname;

    create table meta2 as
    select distinct memname, count(memname) as dervar from meta
    where substr(left(name),3) in ('DECOD' 'HLGT' 'HLT' 'LLT' 'LLTCD' 'PTCD'
                                   'HLTCD' 'HLGTCD' 'SOCCD')
    group by memname
    order by memname;

quit;

data both;
    merge meta1 meta2;
    by memname;
    if orgvar>0 and dervar>0;
run;

proc sql noprint;
    select count(*) into :tot
    from both
    ;
quit;

%let tot=&tot;

** assign these dataset with individully correspoinding macro parameters ;
proc sql noprint;
    select memname
    into :dat1 - :dat&tot
    from both
    ;
quit;

** print out records which --XXX not derived but with non-blank --TERM or --TRT
values  ;
%do i=1 %to &tot ;

    proc sql noprint;
        select name into :orgvar
        from meta
        where memname="&&dat&i" and substr(left(name),3) in ('TERM' 'TRT')
        ;
    quit;

    proc sql noprint;
        select count(*) into :vartot
        from meta
        where memname="&&dat&i" and
              substr(left(name),3) in ('DECOD' 'HLGT' 'HLT' 'LLT' 'LLTCD' 'PTCD'
                                       'HLTCD' 'HLGTCD' 'SOCCD')
        ;
    quit;

    %let vartot=&vartot;
```

```
    ** assign these derived var with individully correspoinding macro parameters ;
    proc sql noprint;
        select name
        into :var1 - :var&vartot
        from meta
        where memname="&&dat&i" and
              substr(left(name),3) in ('DECOD' 'HLGT' 'HLT' 'LLT' 'LLTCD' 'PTCD'
                                       'HLTCD' 'HLGTCD' 'SOCCD')
        ;
    quit;

    %do j=1 %to &vartot;
        data c&j (keep= usubjid &orgvar &&var&j) ;
            set &inlib..&&dat&i;
            if &orgvar ne '' and &&var&j = '';
        run;

        proc sort data=c&j;
            by usubjid &orgvar;
        run;
    %end;

    data _prt;
        retain usubjid &orgvar;
        merge %do j=1 %to &vartot; c&j %end;;
        by usubjid &orgvar;
    run;

    proc print;
        title "&inlib..&&dat&i has the following uncoded --DECOD or high level
               variables ";
    run;

    proc freq;
        title "&inlib..&&dat&i has the following uncoded &orgvar ";
        table &orgvar;
    run;
%end;

%mend chkDerived;

%chkDerived(inlib=adsd );
```