PharmaSUG2010 - Paper CD02

ADaM Dataset Checking Toolkit

Huei-Ling Chen, Merck & Co., Inc., Rahway, NJ

ABSTRACT

An analysis dataset in a clinical trial should be clear and logically unambiguous. The Clinical Data Interchange Standards Consortium (CDISC), a non-profit organization established to define standard data formats to support the exchange of clinical data among and within organizations such as pharmaceutical and medical device companies, contract research organizations, universities, medical research institutions, and regulatory agencies recently provided the ADaM Implementation Guide (ADaMIG), Version 1.0. to specify the preferred features of the Analysis Data Model (ADaM) for a clinical trial. The purpose of this paper is to describe macros to assist in validation of an ADaM dataset. Based on the ADaMIG v1.0, this paper provides six macros as a toolkit to evaluate whether the dataset complies with the CDISC Analysis Data Model.

KEYWORDS

ADaM, CDISC, SUBMIISSION, SASHELP, VMEMBER, VCOLUMN

INTRODUCTION

The ADaM standards were designed to support submission by a sponsor to a regulatory agency. To further help sponsors implement these ADaM standards, the ADaMIG was created. The ADaMIG specifies ADaM standard dataset structures, variables, dataset naming conventions, variable values algorithm, etc. Previously, an analysis dataset was a subjective creation in which variable algorithm and data structures could vary across different protocols, indications, compounds, or sponsors. Now, though ADaMIG allows for flexibility with the dataset, it does have some minimal requirements and some preferred features. Reading through the ADaMIG, this paper summarizes the following minimal requirements and preferred features:

Minimal Requirement	1	Existence of a subject-level analysis dataset ADSL	%ADDATA
	2	Dataset Naming Convention (Basic Data Structure (BDS) dataset should be named using the convention "ADxxxxxx")	
	3	Required Variables for Analysis Datasets	%REQVAR
	4	Value Algorithm (e.g. CHG = AVAL – BASE; PCHG = CHG / BASE)	%CHKVALUE
Preferred (but Not Required) Features	5	One-to-One Correspondence relationship between certain variables, e.g. AVISIT and AVISITN	%ONE2ONE
	6	No Empty Variables in the dataset	%EMPTYVAR
	7	Populate Rule	%POPULATE

TOOLS: SASHELP Dictionary Views

SAS sessions provide Dictionary tables as metadata for the active datasets. Each Dictionary table has an associated PROC SQL view in the SASHELP library. These views are helpful tools when there is a need to look up at the datasets or variables. This paper uses the following two views as the main tool to check the datasets/variables:

SASHELP.VMEMBER: Contains information for all member types about all objects that are in currently defined SAS data

SASHELP.VCOLUMN: Contains information about columns in all known tables

Check 1. EXISTENCE OF A SUBJECT-LEVEL ANALYSIS DATASET ADSL
Check 2. DATASET NAMING CONVENTION

The ADaM basic data structure consists of the following two types of dataset:

- Subject-level dataset ADSL
- Basic Data Structure (BDS)

According to Analysis Data Model Version 2.1, the subject-level analysis dataset will be named "ADSL." The BDS datasets will be named using the convention "ADxxxxxx."

This paper uses VMEMBER from SASHELP library to check if the dataset in the specified analysis dataset folder are in compliance with the naming convention.

Key Syntax for Macro %ADDATA (INLIB =)

Step 1: Use SASHELP.VMEMBER to retrieve the LIBNAME="&INLIB" and MEMNAME="ADSL" record

```
data adslyn;
     set sashelp.vmember(where=(libname="&inlib" and memname="ADSL" and
                                memtype="DATA" ) );
     run;
```

Step 2: If there is no 'ADSL' record from Step 1, issue a warning message

```
proc sql noprint;
     select count(*) into: coln
     from adslyn;

%if &coln = 0 %then %do;
 [ issue warning message here ]
%end;
```

Step 3: Use SASHELP.VMEMBER to retrieve LIBNAME="&INLIB" and MEMTYPE="DATA" record. Check whether the first two characters string of MEMNAME is 'AD' which is the naming convention for ADaM dataset

```
data others;
     set sashelp.vmember(where=(libname="ADAM" and
                                memtype="DATA" ) );
     if substr(memname,1,2) ne "AD" then adflag = 0;
     else adflag = 1;
     run;
```

Check 3. REQUIRED VARIABLES FOR ANALYSIS DATASET

In the section "Standard ADaM Variables," the metadata tables contain a "Core" column that describes whether a variable is required, conditionally required, or permissible. The following table lists the required variables for both the subject level dataset ADSL and the multiple-record-per-subject datasets:

ADSL	THE MULTIPLE-RECORD-PER-SUBJECT DATASETS
STUDYID	STUDYID
USUBJID	USUBJID
SUBJID	TRTP
SITEID	PARAM
AGE	PARAMCD
AGEU	AVAL/AVALC (at
SEX	least one)
RACE	
ARM	
TRTxxP	

There are multiple ways to check the existence of required variables in a dataset. This paper uses VCOLUMN from the SASHELP library to check if the dataset in the specified analysis dataset folder is in compliance with the naming convention.

Key Syntax for Macro %REQVAR (INLIB =, ADAMDATA=)

Step 1: Use SASHELP.VCOLUMN to retrieve variable NAME from the following LIBNAME="&INLIB" and MEMNAME="ADSL" record. This dataset would contain all variables from ADSL as its observations.

```
data cols(keep=name );
     set sashelp.vcolumn(where=(libname="&inlib." and memname="ADSL") ) ;
     run;
```

Step 2: Create a dataset with required variables as the observations.

```
data adslvar;
     STUDYID = .;
     USUBJID = .;
     SITEID  = .;
     AGE     = .;
```

```
     SEX     = .;
     RACE    = .;
     TRTSTDT = .;
     TRTENDT = .;
     run;

proc transpose data= adslvar out= adslvar;
     run;
```

Step 3: Merge dataset from Step1 and Step2. Mark the variable which is from Step 2 dataset but does not exist in Step 1 dataset.

Check 4. VALUE ALGORITHM

The following three items are related to variable value which should be checked and they are:

- If exist, PARAMTYP in ('DERIVED', null value)
- CHG = AVAL – BASE
- PCHG = (AVAL - BASE) / BASE * 100

Key Syntax for Macro %CHKVALUE (INLIB =, ADAMDATA=)

Step 1: Use SASHELP.VCOLUMN to retrieve variable NAME for the checked variable CHG, PCHG, or PARAMTYP. This paper uses CHG as an example to show how to check the algorithm for this variable.

```
     data cols(keep=name );
          set sashelp.vcolumn(where=(libname="&inlib." and memname="&ADaMData." and
                                     name in ('CHG','AVAL,'BASE') )) ;
     run;
```

Step 2: Check the existence of checked variables CHG as well as AVAL and BASE. When variable CHG exists in the dataset, variables AVAL and BASE should also exist in the dataset.

```
     data cols1 cols2 cols3;
          set cols;
          if name = 'CHG'  then output cols1;
          if name = 'AVAL' then output cols2;
          if name = 'BASE' then output cols3;
          run;

     %let dsid=%sysfunc(open(cols1,i));
     %let nobschg=%sysfunc(attrn(&dsid,NOBS ));
     %let rc=%sysfunc(close(&dsid));

     %let dsid=%sysfunc(open(cols2,i));
     %let nobsaval=%sysfunc(attrn(&dsid,NOBS ));
     %let rc=%sysfunc(close(&dsid));

     %let dsid=%sysfunc(open(cols3,i));
     %let nobsbase=%sysfunc(attrn(&dsid,NOBS ));
     %let rc=%sysfunc(close(&dsid));

     %if &nobschg > 0 and ( &nobsaval = 0 or &nobsbase = 0) %then %do;
          proc print data=cols1;
               var name;
               title 'variable AVAL and BASE should be in the dataset with CHG';
               run;
     %end;
```

Step 3: Check that the value of CHG is equal to AVAL minus BASE.

```
   %if &nobschg > 0 and &nobsaval > 0 and &nobsbase > 0 %then %do;
          data chk;
               set &inlib..&adamdata;
```

```
               newchg = aval - base;
               if chg ne newchg;
               run;

          proc print data=chk ;
               var  subjid aval base chg newchg ;
               title 'variable CHG value not equal to AVAL - BASE (NEWCHG)';
               run;
     %end;
```

Check 5. One-to-One CORRESPONDENCE

In the multiple-record-per-subject datasets, some numeric variables are required to be one-to-one mapped to its mapping character variables. Following is a list of the expected one-to-one mapping relation variables:

AVISITN and AVISIT
TRTxxA and TRTxxAN
TRTP and TRTPN
TRTA and TRTAN
TRTSEQP and TRTSEQPN
TRTSEQA and TRTSEQAN
TRTPGy and TRTPGyN
TRTAGy and TRTAGyN
SHIFT and SHIFTC
PARAM and PARAMN and PARAMCD
AVAL and AVALC (AVALC can be a formatted text version of AVAL but if so there should be a one-to-one map between AVAL and AVALC.)

Example of Non-Compliance

The following example shows the non-compliance on the one-to-one mapping rule. PARAMN has two values, 10 and 20,for PARAM values Weight (kg) which should be fixed.

PARAM	PARAMN	PARAMCD	AVISITN	AVAL
Weight (kg)	10	WEIGHT	0	50
Weight (kg)	20	WEIGHT	1	55
Weight (kg)	20	WEIGHT	2	56

Key Syntax for Macro %ONE2ONE (INLIB =, ADAMDATA=)

Step 1: Store the checked variables into two macro parameters (&VARALIST and &VARBLIST). Use a %DO loop to process the checking variable by variable.

```
     %let varalist=trt1a  trt1p  avisit  trta  trtp  trtseqp  trtseqa  aval  shift  ;
     %let varblist=trt1an trt1pn avisitn trtan trtpn trtseqpn trtseqan avalc shiftc ;

     %let i = 1;
     %let vara = %scan(&varalist, &i);
     %let varb = %scan(&varblist, &i);
     %do %while(%length(&vara));

.
.
.

    %end;
```

Step 2: Use SASHELP.VCOLUMN to check the availability of the mapped variables.

```
data cols(keep=name );
     set sashelp.vcolumn(where=(libname="&inlib." and memname="&ADaMData."
                                and lowcase(name) in ("&vara.","&varb.")) );
     run;

proc sql noprint;
     select count(*) into: coln
     from cols;
```

Step 3: Use PROC FREQ and PROC SORT NODUPKEY feature to check the mapping relation. The logic behind the syntax is to compare the count of combinations of the checked variables, e.g. PARAMN and PARAMCD, with the count of each variable's own entries.

Example Dataset:

SUBJID	PARAMN	PARAMCD
1	10	WEIGHT
1	20	WEIGHT
1	20	WEIGHT
1	30	HEIGHT
1	30	HEIGHT

Use PROC FREQ statement to get the combination of PARAMN and PARAMCD.

PARAMN	PARAMCD
10	WEIGHT
20	WEIGHT
30	HEIGHT

The count of the combination of PARAMN and PARAMCD in this dataset is 3.
The count of individual PARAMN is 3.
The count of individual PARAMCD is 2.

When the counts of the combination of tested variables do not equal the counts of each variable, the one-to-one mapping rule is not satisfied.

```
%if &coln = 2 %then %do;
    proc freq data=&inlib..&adamdata noprint;
         tables &vara * &varb / missing list out=out1;
         run;

    data out1a;
         set out1;
         if &vara eq missing and &varb eq missing then delete;
         run;

    proc sort data=out1a out=out1aa nodupkey;
         by &vara;
         run;

    proc sort data=out1a out=out1ab nodupkey;
         by &varb;
         run;

    proc sql noprint;
         select count(*) into: n
         from out1a;

         select count(*) into: n1
         from out1aa;

         select count(*) into: n2
         from out1ab;
         quit;

     %if (&n eq &n1) and (&n eq &n2) %then %do;
          proc print data=out1;
          title "&vara and &varb are one-on-one match";
          run;
     %end;
```

Check 6. EMPTY VARIABLES

For the submitted ADaM dataset, there should not be any null variables which have empty values.

This paper uses variable TYPE in SASHELP.VCOLUMN to assign variables into two groups: Character and Numeric. Use a data step to check the count of the missing value for each variable. When the count of missing values for a particular variable equals the number of records in the dataset, this variable is an empty variable.

Key Syntax for Macro %EMPTYVAR (INLIB=, ADAMDATA=)

Step 1: Use SASHELP.VCOLUMN to retrieve variable NAME and TYPE for the dataset by using this criteria (libname="&inlib" and memname="&adamdata" and memtype="DATA").

```
data adam;
     set sashelp.vcolumn(where=(libname="&inlib" and memname="&adamdata" and
                                memtype="DATA" ) );
     run;
```

Step 2: Use variable TYPE to differentiate variable to character variables, TYPE = 'CHAR', or numeric variables, TYPE = 'NUM'.

```
     proc sql noprint;
          select distinct name
          into :char1 separated by ' '
          from adam
          where upcase(type)='CHAR'
          ;
          select distinct name
          into :num1 separated by ' '
          from adam
          where upcase(type)='NUM'
          ;
          quit;
```

Step 3: Count the number of observations for each character variable with a null value - an empty space in the value. Count the number of observations for each numeric variable with missing value. When the number of empty observations equals the total number of observations, this variable is an empty variable.

```
     %let char2 = &char2._1;
     %let num2 = &num2._1;

     data final;
          set &inlib..&adamdata;
          array char1{*} &char1;
          array char2{*} &char2;
          array num1{*} &num1;
          array num2{*} &num2;
          do i=1 to dim(char1);
             if char1{i} = ' ' then char2{i}=1;
          end;
          do i=1 to dim(num1);
             if num1{i}=. then num2{i}=1;
          end;
          run;

     proc summary data=final;
          var &char2 &num2;
          output out=result
                 n = &char1 &num1;
          run;
```

Check 7. POPULATE RULE

There are two types of populate rules to be checked in this section. Populate refers to the existence of non-null values for a variable.

For the following variables, if used for a given PARAM, this variable should be populated for all records of that PARAM.

BASE
BASEC
BASETYPE
CHG
CHGC
PCHG
PARAMTYP

Variable DTYPE is required if any row in the dataset is derived within a parameter. DTYPE should not be populated for all records within a PARAM because it is impossible that every record is a derived record based on other 'derived' records.

Example 1 of Non-Compliance

USUBJID	PARAM	AVISITN	AVAL	BASE
1001	Weight (kg)	-1	50	49
1001	Weight (kg)	0	49	49
1001	Weight (kg)	2	51	

Variable BASE is used for PARAM Weight (kg) for USUBJID 1001. The value of BASE for AVISITN 2 should not be empty.

Example 2 of Non-Compliance

USUBJID	PARAM	AVISITN	AVAL	DTYPE
1001	Weight (kg)	-1	50	DERIVED
1001	Weight (kg)	0	49	DERIVED
1001	Weight (kg)	2	51	DERIVED

Key Syntax for Macro %POPULATE (INLIB=, ADAMDATA=)

Step 1: Use SASHELP.VCOLUMN to retrieve variable NAME and TYPE for the checked variable BASE, BASEC, BASETYPE, CHG, CHGC, PCHG, and PARAMTYP. Check the existence of these variables.

```
data adam(keep=name type );
     set sashelp.vcolumn(where=(libname="&inlib." and memname="&ADaMData." and
               name in ('BASE', 'BASEC', 'BASETYPE', 'CHG', 'CHGC', 'PCHG', 'PARAMTYP') )) ;
     run;
```

Step 2: If the variable exists and is used for one PARAM, the variable should be populated for all records of that PARAM. Count the number of non-empty records within each PARAM. This count should match with the count of each PARAM.

```
proc sql noprint;
     select count(name) into :tot
     from adam
    ;
     quit;

%let tot=&tot;

proc sql noprint;
     select name into :name1 - :name&tot
     from adam;
     select type into :type1 - :type&tot
     from adam
               ;
     quit;

%do _j=1 %to &tot;
     data final;
     set &adamdata;
     %if &&type&_j = char %then %do;
         if &&name&_j  ne ' ' then check=1;
     %end;
     %else %do;
         if &&name&_j  ne .   then check=1;
     %end;
```

```
     run;

     proc summary data=final;
          var check ;
          class paramn ;
          output out=result
          n = &&name&_j;
          run;

     proc print data=result noobs;
          where paramn ne . and &&name&_j ne _freq_ and &&name&_j ne 0 ;
          var paramn ;
          run;
%end;
```

To check whether variable DTYPE is populated for all records within a PARAM or not, there are two steps. The first step is to check the existence of variable DTYPE. The second step is to check whether DTYPE is populating in all records within a PARAM.

Step 1: Use SASHELP.VCOLUMN to retrieve variable NAME and TYPE for the checked variable DTYPE. Check the existence of this variable.

```
data cols0(keep=name type );
     set sashelp.vcolumn(where=(libname="&inlib." and memname="&ADaMData." and
                                name in ('DTYPE') )) ;
     run;

%let dsid=%sysfunc(open(cols0,i));
%let nobsdtyp=%sysfunc(attrn(&dsid,NOBS ));
%let rc=%sysfunc(close(&dsid));
```

Step 2: DTYPE should not be populating in all records within a PARAM. If there is such a case, a warning message should be issued. When the number of the total observations within each PARAM equals to the number of non empty DTYPE within each PARAM, the DTYPE is populated for all records which should be corrected.

```
          %if &nobsdtyp > 0 %then %do;
               data final;
                    set &adamdata;
                    if dtype  ne ' ' then check=1;
                    run;

               proc summary data=final;
                    var check ;
                    class paramn ;
                    output out=result
                           n = dtype;
                    run;

               proc print data=result noobs;
                    where paramn ne . and dtype eq _freq_ ;
                    var paramn ;
                    title "variable DTYPE in dataset &inlib..&adamdata";
                    title3 "DTYPE should not be populated for all records within a PARAM ";
                    run;

          %end;
```

CONCLUSION

A clear analysis dataset can greatly facilitate the regulatory agency review. ADaMIG is a guideline document to help sponsors prepare an analysis dataset with expected compliance. The macros and syntax provided in this paper assists the project lead or lead programmer in evaluating whether an analysis dataset is compliant with ADaMIG. The checked items listed in this paper are not inclusive of all rules to be followed. Other compliance rules exist which not covered in this paper due to brevity. ADaMIG should be consulted and these additional compliance rules should be examined when preparing an analysis dataset.

REFERENCES

- Analysis Data Model Version 2.0 Final, August 24, 2006 (CDISC website: http://www.cdisc.org/models/adam/V2.0/index.html)
- ADaM Implementation Guide, Version 1.0 (ADaMIG v1.0), May 30, 2008 (CDISC website: http://www.cdisc.org/models/adam/V2.1_Draft/index.html)

ACKNOWLEDGMENTS

The authors would like to thank the management team of Merck Research Laboratories for their advice on this paper/presentation.

Contact Information

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

Huei-Ling Chen
Merck & Co., Inc.
126 Lincoln Avenue
P.O. Box 2000
Rahway, NJ 07065
Phone: 732-594-2249
e-mail: Huei-Ling_Chen@merck.com

TRADEMARK