Paper 199-2013

From SDTM to ADaM

Suwen Li, Everest Research Services Inc., Markham, ON.
Sai Ma, inVentiv Health Clinical, Burlington, ON
Bob Lan, Everest Research Services Inc., Markham, ON.
Regan Li, Hoffmann-La Roche Ltd., Mississauga, ON

ABSTRACT

Clinical Data Interchange standards Consortium (CDISC) Study Data Tabulation Model (SDTM) and Analysis Data Model (ADaM) datasets are designed with highly desirable data and dataset standards for regulatory submission datasets, as referenced in United States (US) Food and Drug Administration (FDA) guidance. Sponsors are increasingly submitting both SDTM and ADaM datasets to regulatory agencies. Whereas SDTM datasets source raw data, ADaM datasets are derived from SDTM datasets and may contain both raw and derived data. The distinctive structure to SDTM datasets brings forth some problems when deriving ADaM data. This paper describes the problems encountered when we derive ADaM data and illustrate how to resolve them using examples.

INTRODUCTION

The main advantage of SDTM data is its standardization. When deriving ADaM data from SDTM data, the standardization provides increased efficiency in the use of standardized programs and codes because the SAS programs are reusable. However, the SDTM data can introduce problems when deriving ADaM data because of SDTM distinguish data structures. In practice, most of time, program preparation starts when data collection is ongoing. Raw datasets may be empty or incomplete with some variables containing null values. Correspondingly, some SDTM datasets are missing or incomplete when some variables in raw CRF data contain null values. Thus, the ADaM programs cannot derive variables from SDTM due to lack of data. Ideally all SDTM domains or records would be available early for programming, as they would be if programming was to occur after all the data been collected and reflected in SDTM datasets. Therefore programmers need to repeatedly check SDTM datasets each time they are updated with new data from the study database so that ADaM programs can then be updated. This time consuming manual process can be error prone if a programmer does not check for updated SDTM datasets which may include additional SDTM data panels or values. To avoid this type of error, when ADaM program preparation occurs while data collection is ongoing, programmers need to consider all situations in advance no matter the domain or values presently exist or not. SUPP-- and CO are the two particular domains that may not exist or may have partial data during early ADaM program preparation. This paper discusses the issues caused by partial SUPP-- or missing SUPP-- and CO and provides a solution with corresponding SAS code.

SUPP DATASET

According to the CDISC SDTM Implementation Guide, sponsors can only add identifier variables, timing variables, and qualifier variables. Any other additional information is included in the Supplemental Qualifiers special-purpose dataset SUPP--. Those records in SUPP-- will be merged back to its corresponding main domain by using the identifying variables IDVAR and IDVARVAL so that new variables from the additional information can be derived correctly in ADaM. However, in practice, not all data are available for programming because program preparation starts when data collection is still ongoing. SUPP-- may only include part of the non-SDTM variables defined in the Case Report Form (CRF). Sometime, SUPP-- may not even exist. When we run the programs developed to create ADaM datasets, the resulting SAS log files may have warning or error messages. Furthermore, the ADaM datasets may be missing variables needed when preparing programs to create analysis output Tables, Figures, and Listings (TFLs). We must consider all situations in advance when deriving ADaM from SDTM datasets to avoid those problems.

In the below example, a CRF collects rash information and its location:

Location of Reaction

Face

Neck

Chest

Other, Specify: _____

In ADaM ADAE dataset, the 4 variables AELOC1, AELOC2, AELOC3, and AELOCOTH are required.

The complete SUPPAE dataset

USUBJID	IDVAR	IDVARVAL	QNAM	QLABEL	QVAL
99-123	AESEQ	6	AELOC1	Location of the Reaction 1	FACE
99-123	AESEQ	6	AELOC2	Location of the Reaction 2	NECK
99-123	AESEQ	6	AELOC3	Location of the Reaction 3	CHEST
99-124	AESEQ	1	AELOC1	Location of the Reaction 1	FACE
99-124	AESEQ	1	AELOC3	Location of the Reaction 3	CHEST
99-124	AESEQ	1	AELOCOTH	Other Location of Reaction	RIGHT ARM

The above SUPPAE dataset includes all of the 3 listed locations and other location. Below SAS codes will merge the SUPPAE back to AE dataset.

```
data suppae;
  set suppae;
  qnam=strip(qnam);
  where idvar='AESEQ';
  aeseq=input(idvarval, best.);
run;

proc sort data=suppae;
  by usubjid aeseq;
run;

proc transpose data=suppae out=suppae2(drop=_name_);
   by usubjid aeseq;
   var qval;
   id qnam;
   idlabel qlabel;
run;

proc sort data=ae;
   by usubjid aeseq;
run;

data ae2;
   merge ae suppae2;
   by usubjid aeseq;
run;

proc print data=suppae2;
run;
```

SUPPAE2 dataset print:

```
Obs    USUBJID    AESEQ    AELOC1    AELOC2    AELOC3    AELOCOTH

 1     99-123       6       FACE      NECK      CHEST
 2     99-124       1       FACE                CHEST     RIGHT ARM
```

However, if the data is not completed during the program preparation stage, for example, the SUPPAE dataset is like below.

USUBJID	IDVAR	IDVARVAL	QNAM	QLABEL	QVAL
99-123	AESEQ	6	AELOC1	Location of the Reaction 1	FACE
99-123	AESEQ	6	AELOC2	Location of the Reaction 2	NECK

SUPPAE2 dataset print:

```
Obs    USUBJID    AESEQ    AELOC1    AELOC2

1     99-123       6       FACE      NECK
```

The dataset generated by above codes only has variables AELOC1 and AELOC2. In the final dataset, when assigning labels and formats to variables, the note “variable is uninitialized” will occur in SAS log file. Sometimes, programs for TFLs cannot be prepared and tested smoothly because of missing variables in dataset. Because AELOC3 and AELOCOTH are specified in CRF, the two variables AELOC3 and AELOCOTH need to be included in ADAE. We can use hard code temporarily or other methods to include all the 4 variables. The drawback is we may forget to remove those hard codes in the final stage when the database is closed or the log file may have variable uninitialized message. In order to derive ADaM ADAE dataset correctly without any other problems, we provide two approaches. The first approach is creating an empty dataset using ATTRIB statement with all variables.

```
data adae;
  attrib
   STUDYID      label='Protocol Number' length=$100
   AELOC1       label=’ Location of the Reaction 1’ length=$20
   AELOC2       label=’ Location of the Reaction 2’ length=$20
   AELOC3       label=’ Location of the Reaction 3’ length=$20
   AELOCOTH     label=’ Other Location of Reaction’ length=$20
    .
    .
   ;
  retain _character_ '';
  stop;
run;
```

Then we concatenate the empty dataset ADAE and AE2. Even if SUPPAE2 has no values for AELOC3 and AELOCOTH in QNAM, the final ADAE dataset will have the two variables with all missing values. The SAS log file will have no variable uninitialized note.

The second approach is using data steps and macro rather than using PROC TRANSPOSE.

```
%macro supp(var=, label=);

  data &var;
    set suppae;
    where qnam="&var" ;
    &var=qval;
    label &var="&label" ;
    keep usubjid aeseq &var;
  run;

  proc sort data=&var;
    by usubjid aeseq;
  run;

%mend;

%supp(var=AELOC1, label=Location of the Reaction 1);
%supp(var=AELOC2, label=Location of the Reaction 2);
%supp(var=AELOC3, label=Location of the Reaction 3);
%supp(var=AELOCOTH, label=Other Location of Reaction);

data suppae2;
   merge aeloc1 aeloc2 aeloc3 aelocoth;
   by usubjid aeseq;
run;
proc print data=suppae2;
run;
```

SUPPAE2 dataset print:

```
  Obs    USUBJID    AESEQ    AELOC1    AELOC2    AELOC3    AELOCOTH

   1     99-123       6       FACE      NECK
```

In these ways, we can derive all the 4 required variables no matter the SUPP-- dataset is completed or not. There is no need to check data and update the program after database close.

Above, we discussed the problem caused by a partial SUPP dataset and provided our resolution. In some studies, we experienced that SUPP may not even exist. In this example, the CRF has a list of races accompanied by "Other, Specify."

Race

American Indian or Alaska Native

Asian

Black or African American

Native Hawaiian or Other Pacific Islander

White

Other, Specify:

_____

The free-text value in the field "Other, Specify" will be placed in SUPPDM dataset. It is very common that only very few subjects have other races that are not in the list. During the programming preparation phase, the SUPPDM may not exist because no subjects have other races. The common mistake made by programmers is ignoring the SUPPDM dataset. The ADSL creation program has no SAS codes to handle SUPPDM because it does not exist. The problem is, when the data collection is completed, one or more subjects may have other races and SUPPDM is generated. Therefore, the ADSL program needs to be updated. The worst case is the programmer does not notice SUPPDM is created and forgets to update ADSL program. To avoid those situations, we must consider SUPPDM in ADSL program no matter whether SUPPDM exists or not. The below codes should be included in ADSL program.

```
%macro supp(rdomain=, var=, label=);
 %let rdomain=%upcase(&rdomain);
 %let var=%upcase(&var);

  proc contents data=work._all_ out=content nodetails noprint     ;
  run;

  proc sql noprint;
     select count(memname) into: n
     from content
     where memname="SUPP&rdomain";
  quit;

 proc sql noprint;
   select distinct length into: length
   from content
   where name='USUBJID';
 quit;

  %if &n=0 %then %do;
     data &var;
        length usubjid $%left(&length) &var $200;
        usubjid='';
        &var='';
        label &var="&label" ;
     run;
  %end;

  %else %do;
    data &var;
       set supp&rdomain;
       where qnam="&var" ;
       &var=qval;
       label &var="&label" ;
       keep usubjid &var;
    run;
  %end;
```

```
   proc sort data=&var;
     by usubjid;
   run;

 %mend;

 %supp(rdomain=DM, var=RACEOTH, label=Other Race);

 proc sort data=dm;
   by usubjid;
run;

data dm2;
  merge dm(in=a) raceoth;
  by usubjid ;
  if a;
run;

 proc print;
 run;
```

The print of DM2 dataset at the early stage of data collection without any subjects having other race.

Obs	USUBJID	AGE	AGEU	SEX	RACE	RACEOTH
1	0010001	42	YEARS	M	WHITE	

The print of DM2 dataset after all data were collected.

Obs	USUBJID	AGE	AGEU	SEX	RACE	RACEOTH
1	0010001	42	YEARS	M	WHITE	
2	0010009	21	YEARS	M	OTHER	EAST INDIAN

CO DATASET

CO is another dataset that always does not exist and causes the same problem described above. In addition, the number of additional columns COVALn in CO dataset will vary depending on how long the comments are. We developed a SAS macro to merge back the CO dataset to its related main domain.

```
%macro co (domain=);
  %let domain=%upcase(&domain);
  proc contents data=work._all_    out=content nodetails noprint     ;
  run;

  proc sql noprint;
    select count(memname) into: n
    from content
    where memname='CO';
  quit;

  proc sql noprint;
    select distinct length into: length
    from content
    where name='USUBJID';
  quit;

  %if &n=0 %then %do;
    data co;
       length coval coval1 coval2 $200 usubjid $%left(&length);
       coval='';
       coval1='';
       coval2='';
       &domain.seq=.;
       usubjid='';
```

```
      label coval='Comment'  coval1='Comment' coval2='Comment';
      keep usubjid &domain.seq coval:;
    run;
  %end;

  %else %do;
    data co;
      set &protlib..co;
      where rdomain="&domain" and idvar="&domain.SEQ";
      &domain.seq=input(idvarval,best.);
      keep usubjid &domain.seq coval:;
   run;
 %end;
 proc sort data=co;
   by usubjid &domain.seq ;
 run;
 %mend;
 %co(  domain=EX);
```

The CO dataset will be merged with EX domain to link comments back to its corresponding parent records.

DELETE EMPTY PERMISSIBLE VARIABLE

According to the CDISC implementation guide, a sponsor is free to drop Permissible variables if no data was collected for them. Those Permissible variables not defined in CRF generally will not be submitted in SDTM domain and ADaM datasets if they have no any values. However, our above approach will include those variables in final ADaM datasets. The last step is to drop those variables if they have no any data. A very simple program used to check all missing COVAL is as follows.

```
data _null_;
  set ex;
  array b(*) _character_;
  call symputx('char', dim(b));
run;

data _null_;
  set ex end=last;
  length dropvar $100;
  array b(*) _character_;
  array bm(*) y1-y&char (&char*0);
  do i=1 to &char;
    if b(i) ne '' then bm(i)=1;
  end;
  if last then do;
    do i=1 to &char;
     if bm(i)=0 and index(upcase(vname(b(i))), 'COVAL')
     then dropvar=compbl(dropvar||' '||vname(b(i)));
    end;
    call symput('drop', dropvar);
  end;
run;
```

The macro variable &DROP will have a list of missing COVAL variables.

CONCLUSION

In the CDISC SDTM Implementation Guide, some empty datasets and permissible variables that contain null values will not be submitted. Thus, some SDTM domains or values in SUPP-- may not exist before all data are received. That will result in programming difficulty when we derive ADaM datasets. We discussed problems caused by relationship datasets SUPP-- and CO and provided out solution with SAS codes. There may be other problems when creating ADaM datasets from SDTM. We welcome further discussion.

CONTACT INFORMATION

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

Suwen Li
Everest Clinical Research Services Inc.
675 Cochrane Drive
Suite 408, East Tower
Markham, Ontario, Canada L3R 0B8
(905) 752-5253
suwen.li@ecrscorp.com