PharmaSUG 2019 - Paper PO-282

Joining the SDTM and the SUPPxx Datasets

David Franklin, IQVIA

ABSTRACT

When creating ADaM datasets from SDTM datasets, programmers often do not understand why we have the supplemental SUPPxx datasets and how to deal with the structure of the data with these datasets. This paper will briefly explain why we have these supplemental datasets, and more importantly, introduce a macro that will bring these to their 'parent' and make their use easier to handle.

INTRODUCTION

Those of us dealing with SDTM datasets have all seen them, those supplemental SUPPxx datasets. But what is their purpose and how do we use the data?

The information that goes into an SDTM dataset is defined using a defined using Study Data Tabulation Model version (this defines the SDTM version) and Study Data Tabulation Model Implementation Guide (which gives guidance to how the SDTM is implemented) documents. There are several versions of each document about, but beware, the group that controls SDTM, CDISC, want to be sure you are using the correct SDTM version with the correct SDTMIG version:

- SDTM V1.1 & SDTMIG V3.1.1
- SDTM V1.2 & SDTMIG V3.1.2
- SDTM v1.3 and SDTMIG v3.1.3
- SDTM v1.4, SDTMIG v3.2 and Associated Persons Implementation Guide (SDTMIG-AP) v1.0

So if you are writing for SDTM v1.3, use the document SDTMIG v3.1.3 – do not mix them up.

Current versions of SDTM that the FDA use are 1.2 and 1.31, so check with the recipient of the SDTM datasets which version they require.

Each version of the SDTM have the content of the parent datasets (those without a prefix of ‘SUPP’) very tightly defined, and variables that are outside those defined are put in a ‘SUPPQUAL’ dataset linked to the parent, e.g. SUPPAE are those variables linked to the AE parent.

Lets look at some what is considered demographic (DM) data collected in a study:

- USUBJID – unique subject identifier
- AGE – age of subject at informed consent
- SEX – sex of subject (controlled terms)
- RACE – race of subject (controlled terms)
- RACEOTH – race of subject if not in controlled terms
- ITTFL – ITT population flag
- SAFFL – Safety population flag

In SDTM, population flags are of interest, particularly if the SDTM is constructed from datasets that are used for analysis (non-ADaM!).

Variables USUBJID, AGE, SEX and RACE are defined in the SDTM DM dataset, but interestingly RACEOTH value is not. There is debate on RACEOTH, whether it should go in SC (Subject Characteristics) but guidance usually is that this going in the SUPPDM dataset as it is directly linked to the variable RACE which is in the DM dataset.

Population variables such as ITTFL and SAFFL go into the SUPPDM dataset per guidance.

So now look at the datasets we are building. For the DM dataset we could have the following example of data:

```
USUBJID AGE    SEX    RACE
001          64   M    WHITE
002          66   F    OTHER
```

This is a horizontal dataset, meaning there is a one record per a certain sort order, in this case, USUBJID. Note that other variables are required in the SDTM DM dataset that are not shown here.

But what about RACEOTH, ITTFL and SAFFL? The structure for the SUPPDM dataset, as is all SUPPxx (supplemental) datasets is vertical in structure and have the following key variables:

- STUDYID – Study Identifier
- RDOMAIN – Related domain
- USUBJID – The unique subject identifier
- IDVAR – Variable which identifies the related records (this is usually the Sequence variable within subject)
- IDVARVAL – The value of IDVAR
- QNAM – The variable name
- QLABEL – The variable label
- QVAL – The data value
- QORIG – The origin (CRF/derived, etc.)
- QEVAL – The evaluator

With the data that was unmapped into the SDTM DM dataset, the following example of SUPPDM data:

```
USUBJID  IDVAR  IDVARVAL  QNAM     QLABEL                 QVAL   QORG     QEVAL
  001                     ITTFL    ITT population flag    Y      DERIVED  SPONSOR
  001                     SAFFL    Safety population flag Y      DERIVED  SPONSOR
  002                     RACEOTH  Race, Other            MAORI  CRF
  002                     ITTFL    ITT population flag    Y      DERIVED  SPONSOR
  002                     SAFFL    Safety population flag Y      DERIVED  SPONSOR
```

STUDYID would be the Study Identifier for the study, and RDOMAIN in this case would be ‘DM’.

A couple of things to note.

The structure is vertical.

The record involving RACEOTH would only appear if there was a value for that variable in the original data.

IDVARVAL and IDVAR are blank as we do not need to go down to a single record within multiple records for a subject.

Under SDTM rules, we could set ITTFL for all subjects by leaving the USUBJID value blank for the ITTFL record – if USUBJID is blank, then all records in the parent domain are set to the ITTFL value.

We are now bring the DM (child) and SUPPQUAL (dataset SUPPDM) to make things easier to work with, maybe to construct an ADaM or do some quick analysis of the SDTM data. The very highly recommended method is is to transpose the SUPPDM dataset, then merge this to the DM dataset, using the following code:

```
*Sort parent data;
proc sort data=dm;
  by usubjid; run;

*Set up SUPPQUAL for joining to parent;
proc sort data=suppdm;
  by usubjid;
proc transpose data=suppdm out=_sdm;
```

```
  by usubjid;
  id qnam;
  idlabel qlabel; *Adds a label to the variable;
  var qval;
run;

*Bring the two together;
data _dm;
  merge dm _sdm (drop=_name_ _label_);
  by usubjid;
run;
```

This will produce a single dataset that we can work with containing the following variables:

- USUBJID
- AGE
- SEX
- RACE
- RACEOTH
- ITTFL
- SAFFL

This is much easier to deal with, but only in the case where there is one record per USUBJID in the parent.

What about if there is multiple as in the case of CM for example? SDTM allows for multiple records per USUBJID and has a sequence variable which is (usually) an incremental counter for each observation within a USUBJID set – CM has CMSEQ, LB has LBSEQ, CM has CMSEQ, just to name a few.

Below is an example of a AE dataset where the LLT and HLT terms are stored in the SUPPAE dataset:

```
Dataset: AE

USUBJID   AESEQ   AETERM   AESTDTC     AEENDTC      AEACN
001         1     TIRED    2017        2017-09      DRUG WITHDRAWN
002         1     WEAK     2016-09     2016-09      DRUG WITHDRAWN
002         2     FALL     2017-10-15  2017-10-15   DOSE NOT CHANGED
```

```
SUPPAE
USUBJID  IDVAR  IDVARVAL  QNAM   QLABEL       QVAL                  QORG      QEVAL
001      AESEQ  1         AEHLT  MedDRA High  ASTHENIC CONDITIONS   Assigned
                          Level Term
001      AESEQ  1         AELLT  MedDRA       TIREDNESS             Assigned
                          Lowest Level
                          Term
002      AESEQ  1         AEHLT  MedDRA High  ASTHENIC CONDITIONS   Assigned
                          Level Term
002      AESEQ  1         AELLT  MedDRA       WEAKNESS              Assigned
                          Lowest Level
                          Term
002      AESEQ  2         AEHLT  MedDRA High  NON-SITE SPECIFIC     Assigned
                          Level Term   INJURIES NEC
002      AESEQ  2         AELLT  MedDRA       FALL                  Assigned
                          Lowest Level
                          Term
```

Bringing the AE and SUPPAE dataset together is similar to that given earlier, except we have another layer of sequence by USUBJUD given by IDVAR (sequence number variable) and IDVARVAL (sequence number variable value). To do this the code would be similar to:

```
*Sort parent data;
proc sort data=ae; by usubjid aeseq; run;

*Set up SUPPQUAL for joining to parent;
data suppae;
  length aeseq 8; set suppae;
  aeseq=input(idvarval,best.);
proc sort data=suppae; by usubjid aeseq;
proc transpose data=suppae out=_sae;
  by usubjid aeseq; id qnam; idlabel qlabel;
  var qval; run;

*Bring the two together;
data _ae;
  merge ae _sae (drop=_name_ _label_);
  by usubjid aeseq;
run;
```

It is important to note that a SUPPQUAL dataset may have different values for IDVAR within the one dataset which may mean multiple passes to join it with its parent.

Occasionally when I review programmers' code, I see the merging of data without the PROC TRANSPOSE step, as shown below:

```
*Sort parent data;
proc sort data=ae;
  by usubjid aeseq;
run;

*Set up SUPPQUAL for joining to parent;
data suppae;
  length aeseq 8;
  set suppae;
  aeseq=input(idvarval,best.);
proc sort data=suppae;
  by usubjid aeseq;
run;

*Bring the two together;
data _ae;
  merge ae suppae (keep=usubjid aeseq qnam qval);
  by usubjid aeseq;
run;
```

This method is not advised as joining horizontal structure with a vertical structure is very highly NOT recommended, and when created can be a multi-record horizontal structure.

THE MACRO

Below is a utility macro that will join the SDTM parent and SUPPQUAL datasets, for an entire directory, so you only have to do this once, then program off the 'combined' datasets throughout the entire ADS/TLF process. Note that the directory of SDTM parent and SUPPQUAL datasets are referenced by libname SDTMO, with the joined datasets referenced in a separate directory libname SDTMW – the two must be different.

```
*-----------------------------------------------------*
 Setup
*-----------------------------------------------------*;
options nofmterr mprint symbolgen mlogic;
libname sdtmO "C:\client\project\DATA\SDTM"
              access=readonly;
libname sdtmW "C:\client\project\DATA\SDTM\COMBINED";

*-----------------------------------------------------*
 Bring in data
*-----------------------------------------------------*;
proc delete data=work._all_;
proc datasets library=work nolist nodetails nowarn
              mt=data kill;
  copy in=sdtmO out=work;
  quit;
run;

*-----------------------------------------------------*
 Build list of datasets where nobs>0
*-----------------------------------------------------*;
data _dat0a;
  length parent $8;
  set sashelp.vtable;
  keep parent memname;
  where libname='WORK' and
        memname=:'SUPP' and nobs>0;
  parent=substr(memname,5);
run;
data _dat0b;
   length parent $8;
   set sashelp.vtable;
   where libname='WORK' and
         memname ^in:('SUPP','_') and nobs>0;
   keep parent;
   parent=memname;
run;
data _dat0c;
   merge _dat0a _dat0b;
   by parent;
proc sql noprint;
   select parent
      into :_dsla separated by ' '
      from _dat0c
      where ^missing(memname);
   select parent
      into :_dslb separated by ' '
      from _dat0c
      where missing(memname);
   quit;
```

```
run;

*-------------------------------------------------*
 With each of the SUPP datasets, transpose into a
 form for merging with partent and merge
*-------------------------------------------------*;
%macro tmp;
  %let i=1;

  *-------------------------------------------------*
   Process each parent and SUPPQUAL dataset in turn
  *-------------------------------------------------*;
  %do %while(%scan(&_dsla,&i) ne );

    *If missing IDVAR values, set to a non-possible
     value;
    data SUPP%scan(&_dsla,&i);
      set SUPP%scan(&_dsla,&i);
      if missing(idvar) then idvar='£';
    run;

    *Get list of unique IDVAR values (a dataset may
     have multiple unique values, including blank);
    proc sql noprint;
      select distinct idvar
        into :_idvar separated by '~'
        from SUPP%scan(&_dsla,&i);
      quit;
    run;

    *Make a copy of the SUPPQUAL dataset;
    data _SUPP%scan(&_dsla,&i)0;
      set SUPP%scan(&_dsla,&i);
    run;

    *Process the SUPPQUAL joining with parent
     by each unique IDVAR values;
    %let j=1;
    %do %while(%scan(&_idvar,&j,%str(~)) ne );

      *if IDVAR in sequence is blank (IDVARVAL will
       also be blank;
      %if %scan(&_idvar,&j,%str(~))=£ %then %do;

        *Take blank IDVAR records from SUPPQUAL;
        proc sort data=_SUPP%scan(&_dsla,&i)0
                  out=_SUPP%scan(&_dsla,&i)0a
                      (drop=idvar idvarval);
          by usubjid qnam;
          where idvar='£';

        *Transpose by subject;
        proc transpose data=_SUPP%scan(&_dsla,&i)0a
                       out=_SUPP%scan(&_dsla,&i)0b;
          by usubjid;
          id qnam;
          idlabel qlabel;
          var qval;

        *Join by subject to parent;
        proc sort data=%scan(&_dsla,&i);
          by usubjid;
        data %scan(&_dsla,&i);
          merge %scan(&_dsla,&i)
```

```
        _supp%scan(&_dsla,&i)0b (drop=_name_);
    by usubjid;
  run;

%end;

*If IDVAR in sequence is non-blank;
%else %do;

  *Determine type of variable in parent for
   IDVAR (char or num) and length;
  data _null_;
    set sashelp.vcolumn;
    where libname='WORK' and
       upcase(memname)=
        "%upcase(%scan(&_dsla,&i))" and
       upcase(name)=
        "%upcase(%scan(&_idvar,&j,%str(~)))";
    call symput('_vvar',strip(type));
    call symput('_vval',strip(put(length,8.)));
  run;

  *Char variable then construct in SUPPQUAL,
   keeping only records related to IDVAR value;
  %if &_vvar=char %then %do;
    data _SUPP%scan(&_dsla,&i)0a;
      length %scan(&_idvar,&j,%str(~))
             $&_vval.;
      set _SUPP%scan(&_dsla,&i)0;
      where idvar="%scan(&_idvar,&j,%str(~))";
      %scan(&_idvar,&j,%str(~))=idvar;
    run;
  %end;

  *Num variable then construct in SUPPQUAL,
   keeping only records related to IDVAR value;
  %if &_vvar=num %then %do;
    data _SUPP%scan(&_dsla,&i)0a;
      length %scan(&_idvar,&j,%str(~))
             &_vval.;
      set _SUPP%scan(&_dsla,&i)0;
      where idvar="%scan(&_idvar,&j,%str(~))";
      %scan(&_idvar,&j,%str(~))=
        input(idvarval,best.);
    run;
  %end;

  *Transpose by subject and IDVAR variable in
   sequence;
  proc sort data=_SUPP%scan(&_dsla,&i)0a
            out=_SUPP%scan(&_dsla,&i)0a;
    by usubjid %scan(&_idvar,&j,%str(~));
  proc transpose data=_SUPP%scan(&_dsla,&i)0a
                 out=_SUPP%scan(&_dsla,&i)0b;
    by usubjid %scan(&_idvar,&j,%str(~));
    id qnam;
    idlabel qlabel;
    var qval;

  *Join by subject and IDVAR variable in
   sequence to parent;
  proc sort data=%scan(&_dsla,&i);
    by usubjid %scan(&_idvar,&j,%str(~));
  data %scan(&_dsla,&i);
```

```
          merge %scan(&_dsla,&i)
                _supp%scan(&_dsla,&i)0b (drop=_name_);
          by usubjid %scan(&_idvar,&j,%str(~));
        run;

      %end;

      *Get next variable in IDVAR variable sequence;
      %let j=%eval(&j+1);

    %end;

    *Get next parent/SUPPQUAL combination in
     sequence;
    %let i=%eval(&i+1);

  %end;

%mend tmp;
%tmp;
run;

*Move data to COMBINED area -- SUPP with parent,
 and those with no SUPP, putting over parent datasets
 only;
proc datasets library=sdtmW nowarn nodetails nolist
              mt=data kill;
 copy in=work out=sdtmW;
   select &_dsla &_dslb;
 quit;
run;

title1 "Contents Listing of Combined SDTM Data";
proc contents data=sdtmw._all_ varnum;
run;
```

To run this code, set up the LIBNAME SDTMO and SDTMW directory references at the top of the program -- it is strongly suggested that a COMBINED directory be created and referenced using LIBNAME SDTMW directly below that reference using libname SDTMO.

One thing to note that although IDVARVAL is character, the code will create a character or numeric merging variable with the parent depending on the type in the parent dataset.

An example setting for a run is:

```
*-------------------------------------------------*
 Setup
*-------------------------------------------------*;
options nofmterr mprint symbolgen mlogic;
libname sdtmO "C:\TUAI\HYDRO\DATA\SDTM"
              access=readonly;
libname sdtmW "C:\TUAI\HYDRO\DATA\SDTM\COMBINED";
```

where directory

C:\TUAI\HYDRO\DATA\SDTM

is the location of the sorce SDTM data (parent with SUPPQUAL datasets), and

C:\TUAI\HYDRO\DATA\SDTM\COMBINED

is the directory for the parent and SUPPQUAL datasets combined. For example, if there are the following datasets in the directory referenced by the SDTMO libname:

- AE
- CM
- DM
- DS
- EX
- LB
- MH
- SUPPAE
- SUPPCM
- SUPPMH
- SV
- TA

then the directory referenced by SDTMW:

- AE
- CM
- DM
- DS
- EX
- LB
- MH
- SV
- TA

CONCLUSION

When creating ADaM datasets from SDTM datasets it is important for programmers to know why there are supplemental SUPPxx datasets and how to deal with the structure of the data with these datasets. Presented as a macro that will bring these to their 'parent' and produce datasets that are of high quality.

CONTACT INFORMATION

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

David Franklin

david.franklin1@iqvia.com