PharmaSUG 2016 - Paper QT19

A Macro to Automatically Flag Baseline in SDTM

Taylor Markway, SCRI Development Innovations, Carrboro, NC

ABSTRACT

The derivation of the baseline flag in SDTM is a good candidate for using a standard macro because 1) SDTM allows for a generic definition of baseline and 2) baseline derivations can be broken down into a few simple steps. The macro presented in this paper translates the steps for derive baseline into a set of macro parameters. It also provides an option to use SDTM to our advantage by automatically determining the correct parameters, instead of manually determining parameters that should be passed to the macro. By leveraging SDTM, the user is only required to provide input and output data sets.

INTRODUCTION

This paper assumes the readers are familiar with SDTM. However, a few key concepts relevant to the macro are discussed, in particular how the SDTM concepts correspond to the macro's parameters. The macro's assumed inputs and outputs will also be covered, followed by a walk-through of the code showing how these concepts are put into action.

Readers who have a solid background in SDTM may want to skip ahead to the %ms_base section.

SDTM AND BASELINE

To give the SDTM concepts some bearing in relation to the baseline derivation we will use the last non-missing result prior to first dose as our baseline definition in this paper.

Since the baseline flag variable of --BLFL belongs in the findings class domains in SDTM, we will assume all SDTM data sets belong to the findings class. Add --BLFL to other domain classes at your own peril.

Our goal now is to relate our generic definition of a baseline to a generic findings class SDTM domain.

A helpful step in solving any problem is to break it down into pieces. In this case, after breaking down our definition, we will want to relate it to the findings domain.

We can break down our definition of the last non-missing result prior to first dose into four parts:

1. The last … prior to
2. Non-missing
3. Result
4. First dose

The first part "The last … prior to" tells us we are going to need to deal with timing concepts to derive the baseline. Fortunately, the SDTM model makes this simple and has defined timing variables for all domain classes. These timing variables are the variables such as: VISIT, VISITNUM, EPOCH, --DTC, --STDTC, --ENDTC, and so on.

The full list can be found in Section 2.2.5: Timing Variables for All Classes in v1.4 of the SDTM.

As we care about the order, "The last… prior to," we need to ignore any variable that could not be in chronological order when sorted. So we ignore variables such as VISIT, EPOCH, and –TPT. We now have a list of generic timing variables that can be used to determine "The last… prior to."

Part two, "Non-missing" is a specific rule about a specific variable. In this case, it is the result variable. So now we can relate this back to our generic SDTM domain with --ORRES.

Relating this back to a generic SDTM domain is important because we can have any rule here that is equally as valid. For example, we can have a requirement that for any record to be considered for baseline that the evaluator is the adjudication committee (--EVALID="ADJUDICATION COMMITTEE") or that the subject was fasting (--FAST="Y"). A rule is a very specific set of instructions, so we can only relate a specific rule to the generic SDTM findings domain. You can see an example of this in the code section.

Now we come to the "Result," which you may think, is easy, as the result variable is just --ORRES for the findings domains. This is partially right. For a generic findings domain, there could be multiple tests that we need to identify, which would require more information than just the --ORRES variable.

In terms of generic SDTM variables we need to look at three of the five major roles: Identifiers, Topic, and Qualifiers. Identifiers are variables such as STUDYID, USUBJID and DOMAIN. The Topic variable specifies the focus of an observation and, in the findings domains, is --TESTCD. The qualifier variables we are most likely to be interested in are the grouping qualifiers --CAT and --SCAT. To keep things simple, we will refer to this whole set of variables as grouping variables.

Lastly, we have "first dose." The baseline has to be relative to something and, in our case, it is the first dose. In the SDTM the first dose is typically represented in the demographics domain as RFXSTDTC. Alternatives could be the reference start date, RFSTDTC, or perhaps even before a particular visit. This is a rule, just like the second part, and must be translated into a generic set of conditions for a findings domain for our macro to work.

%MS_BASE

The macro we created to derive the baseline is named %ms_base. Before walking through the code we will go over a few key assumptions, the inputs required, and the output.

ASSUMPTIONS

The macro assumes it is provided with an SDTM findings data set. This means it should be used as late in the program as possible after all other SDTM variables have been derived. At a minimum any SDTM variables used in the creation of the baseline flag should be derived.

INPUT

The macro has nine input parameters, and only two are necessary from the user. The input and output data set. The nine parameters are:

1. Dsn –Input data set.
2. Dsout –Output data set.
3. Auto – An indicator variable to automatically determine standard input, if not "YES" then the next set of variables must be provided by the user.
4. XX – SDTM domain.
5. Chrono – Timing variables to sort data to determine the latest record for each group.
6. Groupby – Non-timing variables that identify an individual record.
7. Date – Date to use to determine if a record is prior to the first dose.
8. Result – Character variable that contains the result.
9. Rule – Add a rule that must be met for a record to be considered as a baseline.

There is also one local macro variable defined in %ms_base, RDTC. This is set to the appropriate reference date from the SDTM DM domain. Typically this would be RFXSTDTC, but it is included as a macro variable in case it needs to be changed. The rule parameter can be used in both the automatic and the manual setting. No rules will be applied automatically unless they have been programmed into the macro as defaults.

A sample manual call with a rule to exclude any records with a time point of "POST DOSE" for the EG domain would look like the following:

```
%ms_base(
    dsn=eg,
    dsout=egbl,
    auto=No,
    XX=EG,
    Chrono=EGDTC EGTPTNUM VISITNUM,
    GroupBy=USUBJID EGCAT EGTESTCD,
    Date=EGDTC,
    Result=EGORRES,
    Rule=%str(EGTPT ne ‘POST DOSE’)
    );
```

A sample automatic call for the EG domain with the same rule looks as follows:

```
%ms_base(
    Dsn=eg,
    Dsout=egbl,
    Auto=YES,
    Rule=%str(EGTPT ne ‘POST DOSE’)
    );
```

OUTPUT

The output from %ms_base is identical to the input, but with two exceptions. The --BLFL is now populated with ‘Y’ as determined by the input and, if required, the RDTC is merged from the demographics domain if it did not already exist in the input data set.

The macro also performs a check of the baseline to ensure that the input parameters provided (or automatically determined) uniquely identify one record per combination of the timing and grouping input, CHRONO and GROUPBY. A note will be output to the log if this is not the case.

CODE

Detailed walkthrough of the main parts of the code. This is the full code of the macro broken into sections with additional commentary provided. Slight modifications to the validated version have been made to improve the readability.

Start of Macro and Automatic Determination

The macro starts by the assignment of the RDTC value. Next, if the automatic input determination is indicated, it will start to determine the correct input based on the SDTM assumptions.

```
%macro ms_base(xx=,dsn=,dsout=,auto=YES,date=,rule=,result= groupby=,chrono=);

*--- Assign reference date for base for the study ---*;
%let rdtc=RFXSTDTC;

%if &auto ne YES %then %goto getbase;
*---------------------------------------------------------------*;
*---- Automatically select variables to determine baseline ----*;
*---------------------------------------------------------------*;
*--- Determine domain if not provided ----*;
%if "&xx."="" and %mu_varexist(ds=&dsn.,var=DOMAIN) %then
  %do;
   data _null_;
     set &dsn. (obs=1 where=(domain ne ''));
     call sympust('xx',strip(upcase(domain));
   run;
  %end;
%else
  %do;
    data _null_;
      put "NOTE: &sysmacroname Requires Domain or that XX be provided";
```

```
        put 'NOTE: &sysmacroname will exit without running';
      run;
      %goto exit;
    %end;
```

The macro makes extensive use of the utility macro %mu_varexist, which returns a 1 or 0 if the indicated variable exists in the data set or not. This is used to check the assumptions of the macro and provide helpful feedback to the log if the assumptions are not met.

Automatically Determine CHRONO, GROUPBY, RESULT and DATE

To determine the grouping and timing variables, %ms_base uses a data set that contains the list of possible candidate variables for the timing (CHRONO), grouping (GROUPBY) and the result. Each of the variables in this data set is also associated with a ranking variable. This ranking variable determines the order of the variables in the by statement that is used to sort the data. The date is determined by taking the last date variable from the timing candidates.

```
*--- Create data set containing grouping and timing variables ---*;
*-- This list is then used to extract from the input data set -*;
*-- ORD is the order the variable appear in the by statement -*;
data autolist;
  length NAME GROUP $8;

  GROUP='GROUP';
  NAME="USUBJID";
  ORD=0;
  output;

  NAME="&xx.CAT";
  ORD=1;
  output;

  NAME="&xx.SCAT";
  ORD=2;
  output;

  NAME="&xx.TESTCD";
  ORD=3;
  output;

  GROUP="TIMING";
  NAME="&xx.STDTC";
  ORD=1;
  output;

  NAME="&xx.DTC";
  ord=2;
  output;

  NAME="&xx.ENDTC";
  ord=3;
  output;

  NAME="&xx.TPTNUM";
  ord=4;
  output;

  NAME="VISITNUM";
  ORD=5;
  output;

  GROUP='RESULT';
  NAME="&xx.ORRES";
  output;
```

```
run;
```

Notice that &xx. is used in the variable names so that they will match the domain variable names for any given domain. Also note we only consider the --ORRES as a result variable. With the rules of SDTM this is all that is needed for our baseline definition.

Next, the macro will use the CONENTS procedure to determine the variable in the input data set. The PROC CONTENTS output is then merged with the autolist data set to create a list of variables that should be considered for the baseline determination.

```
*Get list of variables in the input data set *;
proc contents data=&dsn. out=vars noprint;
run;

*prepare variable list to check against autolist*;
proc sort data=vars out=vars;
  by name;
run;

proc sort data=autolist;
  by name;
run;

*Merge the variable list to autolist to determine which variables *;
* Will be used to calcluate baseline *;
data vars;
  merge vars (in=in_vars) autolist (in=in_autos);
  by name;

  if in_vars and in_autos then output;
run;
```

After we have the list of variables that fit the SDTM model to be considered for CHRONO, GROUPBY, and RESULT, we need to convert them into local macro variables. When the macro converts this data to macro variables, then we can handle user provided input and automatically determined input in the same way.

To do this, we need to first sort the data into the groups. Also at this time we will sort by the order variable ORD to allow us to determine which date variable was the last in the list. To have the macro variables CHRONO and GROUPBY mirror the user input, we first concatenate all of the variables into a list before assigning the value to a macro variable with call symputx.

```
*Use the matching auto-variables to populate the baseline macro variables *;
proc sort data=vars;
  by group ord;
run;

data _null_;
  length groupby chrono $200.;
  set vars;
  by group ord;

  retain groupby chrono;

  if group='GROUP' then groupby=catx(' ', groupby, name);
  else if group='TIMING' then chrono=catx(' ',chrono, name);

  if last.group then
    do;
      if group='GROUP' then call symputx('groupby',groupby);
      else if group='TIMING' then call symputx('chrono',chrono);
    end;
  if first.group then
    do;
      if group='RESULT' then call symputx('result',name);
      if group='TIMING' then call symputx('date',name);
    end;
run;
```

Now that we have the automatic determination of the input, we can start the baseline derivation. However, before jumping into the derivation, we include an output to the log to help the user to see what the macro determined as the appropriate input.

```
  *---------------------------------------------------------------*;
  *---- Output summary of automatic determination ----*;
  *---------------------------------------------------------------*;
  data _null_;
    put '======================================================================';
    put '========= ms_base automatic determination of input ===================';
    put '======================================================================';
    put "= GROUPBY: &groupby";
    put "= CHRONO : &chrono.";
    put "= RESULT : &result.";
    put "= DATE   : &date.";
    put "= RULE   : &rule.";
    put '======================================================================';
    put "= ms_base will flag the last non-missing &result. prior to &rdtc.";
    put "= Records prior to &rdtc. will be determined by &date.";
    put "= A &xx.BLFL will be created for each combination of &groupby.";
    put "= The last record prior to &rdtc. will be determined by &chrono.";
    put '======================================================================';
    put '======================================================================';
  run;
```

Determine the Baseline Record

At this point in the code, we are ready to start the derivation of the baseline. The first thing we do is to check the input data set to confirm that it has the RDTC variable. If not,we will merge the DM domain to the input data set so that baseline can be determined.

```
  *---------------------------------------------------------------*;
  *---- Derivations to get baseline ----*;
  *---------------------------------------------------------------*;

  * Check if the reference date is in the data set - add it if needed *;
  %if %eval(%mu_varexist(ds=&dsn.,var=&rdtc.)=0) %then
    %do;
      data _null_;
        put "NOTE: Merging p_sdtm.dm to &dsn. to obtain &rdtc.";
      run;

      *---Getting the reference start date from the demographics---*;
      proc sql noprint;
        create table &dsout. as
          select xx.*, dm.&rdtc.
            from &dsn. as xx left join p_sdtm.dm as dm
              on dm.usubjid=xx.usubjid
        ;
      quit;
    %end;
```

It is also at this point in the macro where adding global rules can take place as they will be used for both the automatic and manual determination. Below is an example of including the --TPT ne ‘POST DOSE’ for including a record as a possible baseline. Note that we have to check that --TPT exists in the data set prior to applying the rule or we could get errors.

```
*--- Automatically check criteria for additional rules ----*;
%local rule1;
%if %mu_varexist(ds=&dsn.,var=&xx.TPT) %then
  %do;
    %let rule1=%str( index(compress((upcase(&xx.TPT),'','AIK')),'POSTDOSE')=0);
  %end;

%if "&rule1" ne "" %then %let rule=(&rule.) and (&rule1);
```

The first step to determine the baseline is to decide if it meets all of our rules. Recall that our definition of baseline is "The last non-missing record prior to first dose." If a record meets the basic criteria to be considered as baseline we put it into a pretreatment category, "1PRE."

```
*---------------------------------------------------------------------------*;
*-            Determine if a record is pre/post reference date            -*;
*---------------------------------------------------------------------------*;
data &dsout.;
  set &dsout.;

  length baseref $5.;

  *Find the minimum resolution for each date*;
  reflen=length(&rdtc.);
  datlen=length(&date.);
  reslen=min(reflen,datlen);

  *Put observations into pre or post baseline buckets *;
  if &rdtc ne '' and &result. ne ''
       and substr(&date.,1,reslen) <= substr(&rdtc.,1,reslen)
       and (&rule.)
       then baseref='1PRE';
  else baseref='2POST';

run;
```

After putting the records into the pre-baseline group, we need to find the latest record in each group. This is done by sorting the data set by the pre- and post-baseline, followed by the grouping variables. Then finally, within each group we sort by the timing variables (CHRONO) to get the last record.

```
*----------------------------------------------------------------------------*;
*---   Determine the last record in the pre bucket and flag as baseline  ---*;
*----------------------------------------------------------------------------*;
*Get last sorting variable to break into categories*;
%let lastsort=%sysfunc(scan(&groupby.,-1));

proc sort data=&dsout.;
  by baseref &groupby. &chrono.;
run;

data &dsout.;
  set &dsout.;
  by baseref &groupby. &chrono.;

  *Flag the last record in the PRE base category as baseline*;
  if last.&lastsort. and baseref='1PRE' then &xx.BLFL='Y';
run;
```

Now that we have flagged the baseline, we want to check that it is a unique record. To do this we check how many records are identified by the combination of GROUPBY and CHRONO. If there is more than one record, a note will be put to the log.

```
*---- Check that there is a unique record for baseline -----*;
proc freq data=&dsout. (where=(baseref='1PRE')) noprint;
  table %sysfunc(tranwrd(&groupby.,%str( ),*))
        *%sysfunc(tranwrd(&chrono.,%str( ),*)) / out=keycheck (drop=percent);
run;

data _null_;
  set keycheck;
  if count >1 then
    do;
      put 'NOTE: Sorting is not unique see keycheck data set with count > 1 ';
    end;
run;

%exit:;
%mend ms_base;
```

This is the end of the macro and we have the input dataset now with an appropriately flagged baseline. There were some modifications to help the code flow better in the paper format, but you can copy and paste each code snippet in the order presented to reproduce the macro.

CONCLUSION

The macro presented here is just one possible incarnation of taking advantage of the SDTM standards to obtain a generic baseline.

REFERENCES

Study Data Tabulation Model (SDTM) Version 1.4

SDTM Implementation Guide (SDTMIG) Version 3.1.2

ACKNOWLEDGMENTS

I would like to thank Jitesh Tiwari, Kiranmai Byrichetti, and Aditya Rachoori for their comments and input during the macro validation process.

CONTACT INFORMATION

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

Name: Taylor Markway
Enterprise: SCRI Development Innovations
Address: 3322 West End Avenue, Suite 900
City, State ZIP: Nashville, TN 37203
Work Phone: (615) 329-7274
E-mail: Taylor.Markway@SCRI-Innovations.com