PharmaSUG 2021 - Paper AP-041

Laboratory Results in Relation to Concomitant Medications (CMs)

Scott Horton, formerly of United BioSource LLC

ABSTRACT

In addition to reviewing laboratory (lab) data based on study treatments, clinicians are also interested in what concomitant medications (CMs) a subject might be taking or might have taken prior to such lab data. This review can be accomplished by the dates in a lab data listing and the dates in a CM listing—this type of review by its very nature is tedious.

Combining lab data and CM data is complex when not limiting the CMs and their dosages being considered for such a clinical review.

A recursive SAS® macro will be examined that simplifies the preparation of CM data to a format that allows combining with lab data in a simple, straightforward manner. The resulting data set contains records of CMs for any given time point for each patient and allows the two types of data to be easily combined for clinical review in listings or tables.

INTRODUCTION

Displaying or summarization lab data by study treatments is very straight forward. Almost all lab listings and tables use assigned or actual treatment. Even for tables and listings that are sensitive to the treatment a subject is on at the time of obtaining lab data (or has taken previously) in an unexpected manner that is not part of the study design can still be easily handled. However, applying these same concepts to CM data where an unknown population of medications and dosages might be encountered removes any sense of simplicity in manipulating the data to get the desired result. To avoid very complex and unwieldly SAS code, the manipulation and reformatting of CM data will be approached in a piecemeal fashion—like eating an elephant one bite at a time.

The resulting CM data will be restructured to allow a simple merge to occur with lab data to produce tables and listings that a clinician will love.

SUMMARIZING LABORATORY RESULTS VERSUS STUDY TREATMENTS

When summarizing lab results based on the treatments in the study design, different approaches can be used. A table can be produced where statistics are computed by study treatment (or study treatment doses) by either 1) the treatment dose groups subjects were randomized to or 2) treatment dose groups received by subjects. Figures can be produced by 1) plotting data in a single panel where treatment dose groups are plotted on the left Y-axis and lab data are plotted on the right Y-axis or 2) plotting data in two panels stacked on top of each other (thus visually sharing the common X-axis of time) but having their own Y-axis (treatment dose group and lab data, respectively).

CONCOMITANT MEDICATIONS INSTEAD OF STUDY TREATMENTS

Let's take some of the same ideas that we use in handling lab data based on study treatments but complicate it to a tremendous degree by displaying and summarizing lab data based on CMs. CM data are usually not restricted to a manageable number of possibilities and doses. This exponential increase in complexity make standard code for handling the data become, well, impossible; especially if dosage of the CMs must be considered.

CONCOMITANT MEDICATIONS EXAMPLE DATA

We will look at seven different CMs with five of those medications where dosage is also considered (to limit the discussion for the purposes of this paper). We will consider the same CM with a different dose as a "different" CM.

First, we need to convert the dosage to a daily dosage. For example, a BID dose will be double and a QOD dose will be halved to obtain daily dose amount. Different dose amounts are not really unlimited, but can be large; and for the purposes of programming, we need to essentially cover an unlimited number of “different” CMs based on either or both the medications and dose (if dose is being considered).

An example of the type of output we are shooting for is found in **Listing 1. Example Listing of CMs Taken in Relation to Laboratory Data**. A second example is found in **Table 1. Example Table of CMs Taken in Relation to Laboratory Data**.

Patient	Medication Combination	Start Date	Stop Date	Visit Date / Study Day	Visit	ALT: (U/L)	xULN	AST: (U/L)	xULN
XXX/00	CM#1 UNKmg + CM#2 UNKmg + CM#3 UNKmg + CM#4 UNKmg + CM#5	2012	2012						
XXX/00				11MAR2013/-31		21		64	<3x
XXX/00	CM#2 20mg	11APR2013	10JAN2016	29APR2013/19	Month 0.5	21		28	
XXX/00	CM#2 20mg	11APR2013	10JAN2016	14JUN2013/65	Month 2	30		43	
XXX/00	CM#2 20mg	11APR2013	10JAN2016	01AUG2013/113	Month 4	29		35	
XXX/00	CM#2 20mg	11APR2013	10JAN2016	11OCT2013/184	Month 6	35		32	
XXX/00	CM#2 20mg	11APR2013	10JAN2016	23DEC2013/257	Month 8	60		62	<3x
XXX/00	CM#2 20mg	11APR2013	10JAN2016	06MAR2014/330	Month 11	47		117	<3x
XXX/00	CM#2 20mg	11APR2013	10JAN2016	06JUN2014/422	Month 15	31		54	
XXX/00	CM#2 20mg	11APR2013	10JAN2016	09SEP2014/517	Month 18	36		34	
XXX/00	CM#2 20mg	11APR2013	10JAN2016	18MAR2015/707	Month 24	29		62	<3x
XXX/00	CM#2 20mg	11APR2013	10JAN2016	15JUN2015/796	Month 27	27		63	<3x
XXX/00	CM#2 20mg	11APR2013	10JAN2016	08JAN2016/1003	Month 33	34		71	<3x
XXX/00	CM#2 20mg + CM#5	11JAN2016		05AUG2016/1213	Month 39	26		26	
XXX/00	CM#2 20mg + CM#5	11JAN2016		03MAR2017/1423	Month 48	23		28	
XXX/00	CM#2 20mg + CM#5	11JAN2016		02JUN2017/1514	Month 51	25		26	
XXX/00	CM#2 20mg + CM#5	11JAN2016		06OCT2017/1640	Month 54	23		19	

Listing 1. Example Listing of CMs Taken in Relation to Laboratory Data

Medication Combination	Total number of patients with this combination Full Analysis Set n (%)	 within this subgroup n (%)	AST ALT ≥3x ULN n (%)	AST/ALT ≥3x ULN n (%)	 both missing n (%)
Any combination	184 (99.5)	61 (100.0)	3 (4.9)	6 (9.8)	2 (3.3)
CM#1 5mg	2 (1.1)	2 (3.3)	0	0	0
CM#1 5mg + CM#2 10mg	1 (0.5)	0			
CM#1 10mg	12 (6.5)	4 (6.6)	0	0	0
CM#1 10mg + CM#3 UNKmg + CM#5	1 (0.5)	0			
CM#1 10mg + CM#4 20mg + CM#2 10mg + CM#5	1 (0.5)	0			
CM#1 10mg + CM#2 80mg	1 (0.5)	0			
CM#1 10mg + CM#6	1 (0.5)	0			
CM#1 10mg + CM#5	3 (1.6)	1 (1.6)	0	1 (100.0)	0
CM#1 20mg	13 (7.0)	1 (1.6)	0	0	1 (100.0)
CM#1 20mg + CM#3 20mg	1 (0.5)	0			
CM#1 20mg + CM#3 20mg + CM#2 10mg	1 (0.5)	0			
CM#2 ...					

Table 1. Example Table of CMs Taken in Relation to Laboratory Data

MATCHING CMS AND LABORATORY RESULTS

In order to produce data that easily lends itself to creative output similar to the two examples just above, we need to restructure the CM data so that each record for a subject contains all of the CMs that are currently being taken.

DIFFERENT SCENARIOS TO CONSIDER

The CMs for each subject fall into different categories. We could divide the data into four groups:

1. No overlaps occur between any CMs for a subject
2. Overlap occurs between only two CMs for a subject at any given time. An example of this is:
    a. CM#1 alone
    b. CM#1 and CM#2 overlap (as CM#2 started after CM#1)
    c. One of the following
        i. CM#1 alone (extends beyond the stop date of CM#2)
        ii. CM#2 alone (extends beyond the stop date of CM#1)
        iii. CM#1 and CM#2 stop on the same date or are both ongoing
3. Overlap occurs between three CMs
4. Overlap occurs between four or more CMs
    a. This is where the code to handle such complex and unknown data lead to solving the problem with a non-standard approach.

"EATING THE ELEPHANT ONE BITE AT A TIME"

Instead of trying to solve this now complex data-handling problem all at once, we need to just address up to two CMs at a time.

Process the first CM (or first two or more CMs if overlap occurs) and output a "new" concomitant record for that portion of the CM data.

Keep the rest of the data that is not output potentially including information from the records used to create the "new" CM mentioned above for further processing. This allows the data to be processed in bite-size portions that are not overwhelming.

Listing 2. CMs Data Before and After Processing shows the CM data prior to manipulation and post manipulation – where the records reflect the combination of medications being taken at any given time on a single record. The dose of 10mg is highlighted in red to denote that even though CM#1 appears twice in the *BEFORE* data, we are considering these medications as unique due to the different dosages.

BEFORE

CM#1	20mg	Start: 01Jan2015	End: 28Nov2017
CM#2	40mg	Start: 01Aug2016	End: 31Dec2016
CM#3		Start: 10Sep2016	End: 30Nov2016
CM#4	100mg	Start: 01Oct2016	End: <ongoing>
CM#1	10mg	Start: 01May2018	End: 31Jul2018

AFTER

CM#1(20mg)	Start: 01Jan2015	End: 31Jul2016
CM#1(20mg) + CM#2(40mg)	Start: 01Aug2016	End: 09Sep2016
CM#1(20mg) + CM#2(40mg) + CM#3	Start: 10Sep2016	End: 30Sep2016
CM#1(20mg) + CM#2(40mg) + CM#3 + CM#4(100mg)	Start: 01Oct2016	End: 30Nov2016
CM#1(20mg) + CM#2(40mg) + CM#4(100mg)	Start: 01Dec2016	End: 31Dec2016
CM#1(20mg) + CM#4(100mg)	Start: 01Jan2017	End: 28Nov2017
CM#4(100mg)	Start: 29Nov2017	End: 30Apr2018
CM#1(10mg) + CM#4(100mg)	Start: 01May2018	End: 31Jul2018
CM#4(100mg)	Start: 01Aug2018	End: <ongoing>

Listing 2. CMs Data Before and After Processing

RECURSIVE MACRO

A recursive macro is a macro that can call itself during its execution; this can be done repeatedly. If it does call itself, the current execution of the macro is interrupted/suspended until the subsequent macro call is finished executing. This pattern is followed until the original macro call finishes execution.

RECURSIVE MACRO INTERSECT

GOAL AND INITIAL CODE OF MACRO INTERSECT

Macro intersect is designed with two important and related features:

1. To only process the first two CMs for each subject
2. To call itself until further processing of CM data is no longer required.

Here is the code from the top of the macro:

```
%macro intersect(data=);
   *** NOTE: parameter &data. appears in both DATA and SET statements ***;
   data &data.(drop = medcomb astdt aendt)
    newfinished(keep = usubjid trtsdt trtedt medcomb astdt aendt mednum);
      set &data. end=last;
         *** ARRAYS WITH CM DATA ***;
         *** CMs STORED IN ARRAYS IN ORDER OF START DATE/END DATE ***;
      array med{&maxmeds.} med1-med&maxmeds.;
      array startdt{&maxmeds.} start1-start&maxmeds.;
      array stopdt{&maxmeds.} stop1-stop&maxmeds.;
…
```

The entire macro will be included later in the paper. The discussion of the algorithm that is reflected in the macro will be described in text first.

PREPARATION OF DATA PRIOR TO INITIAL CALL OF MACRO INTERSECT

The CM data needs to be changed in the following manner in preparation for processing by macro intersect.

1. Store all CMs (of interest if you are limiting to specified subset) on a single record for each subject
    a. Medication (including dosage if desired)
    b. Start Date
        i. Partial date: an imputed start date is used
        ii. Missing date: a date prior to any potential study data is used
    c. Stop Date
        i. Partial date: an imputed start date is used
        ii. Missing date: a date after all potential study data is used
2. Use arrays to simplify the code and store the CM data sorted by Start Date and Stop Date. For example:
    a. MED1: First CM taken
    b. START1: First CM start date
    c. STOP1: First CM stop date

INITIAL DATA STEP IN MACRO INTERSECT

The initial data step does the heavy lifting when processing CM data. Four different scenarios will be discussed.

Patient took only one CM

The CM data for such subjects would only populate the first element in all three arrays (e.g., medication name, medication start date, medication stop date). The single record output to data set NEWFINISHED contains this same information. This record is not kept in the data for further processing (sorting and subsequent inclusion in a recursive call of macro intersect).

If there is data left to process (i.e. subjects had data that did not fall into the Patient took only one CM category), a new call of macro intersect occurs with remaining data.

Patient took two or more CMs

1st CM started and ended before 2nd CM started

Output a record to data set NEWFINISHED that has its CM as the 1st CM with its corresponding start and stop dates; that is, this record reflects the 1st CM's data.

Move the data in the three arrays to the "left" since first elements in three arrays have been processed and are no longer needed (do this in a loop). This overwrites the data that was just output to data set NEWFINISHED since it is no longer needed for further processing.

Decrement number of medications this subject has (e.g., if 4, then now 3 as the 1st CM has been fully processed). The number of medications per subject is being tracked so that we do not have to process empty elements of arrays and so we know when we are only left with a single, unprocessed CM for each subject.

Keep record for further processing in a subsequent macro call. Subject still has unprocessed CMs in arrays.

1st CM started before 2nd CM started but also overlaps 2nd CM

Output a record to data set NEWFINISHED where its CM consists of the 1st CM with its corresponding start date and is assigned a stop date of the day before the 2nd (and overlapping) CM starts.

CM data in first element of each array changes from original data:

1. CM becomes the concatenation of the original 1st CM and original 2nd CM
2. Start Date assign start date of 2nd CM
3. Stop Date *(of this combined CM)*
    a. If original 1st CM stopped on same date as original 2nd CM
        i. If 3 or more CMs, move array elements to "left" with exception of overwriting 1st elements (which was just modified above) – Stop date remains unchanged
        ii. Decrement number of medications this subject has
    b. Else if original 1st CM stopped after original 2nd CM stopped
        i. Stop date of new 1st *combined* CM in array assigned original 2nd CM stop date
        ii. New 2nd CM medication element assigned 1st original CM
        iii. New 2nd CM start date element assigned day after 2nd original CM stop date
        iv. New 2nd CM stop date element assigned 1st original CM stop date
    c. Else original 1st CM stopped before original 2nd CM stopped
        i. New 2nd CM start date element assigned day after original 1st CM stop date

1st CM started on the same date as the 2nd CM

New CM is the concatenation of 1st CM and 2nd CM with the start date of the 1st CM. The new CM is not written out to data set NEWFINISHED at this point since we need to determine if a 3rd CM also started on that same date or overlaps with it. To determine the stop date of this new CM:

1) If original 1st CM stopped on same date as original 2nd CM
    a. If 3 or more CMs, move array elements to "left" with exception of overwriting 1st element (which was just modified above) – Stop date remains unchanged
    b. Decrement number of medications this subject has
2) Else if original 1st CM stopped after original 2nd CM stopped
    a. Stop date of 1st *combined* CM in array assigned 2nd CM stop date
    b. New 2nd CM medication element assigned 1st original CM
        i. As this new 2nd CM (same as original 1st CM) extends beyond the stop date of the original 2nd CM
    c. New 2nd CM start date element assigned day after 2nd original CM stop date
    d. New 2nd CM stop date element assigned 1st original CM stop date
3) Else original 1st CM stopped before original 2nd CM stopped
    a. New 2nd CM start date element assigned day after original 1st CM stop date

Summary of initial data step of macro intersect

Essentially, you see if the data for the 1st CM:

1) Can be output without change
    a. Only CM for subject
    b. Only CM for subject in a subsequent macro call since all other CM information has already been processed and output in prior macro calls
2) Can be output with only 1st CM information
    a. 1st CM stops before 2nd CM starts
    b. Portion of 1st CM occurring prior to start of 2nd CM
        i. Keep portion when CM overlap begins for processing in subsequent macro call
3) Can be output with 2nd CM information
    a. 1st and 2nd CM start on the same date
    b. Combine CM
    c. Determine stop date
    d. Keep for processing in subsequent macro call
        i. Might overlap with 3rd or later CMs

PROCESING DATA AFTER INITIAL DATA STEP IN MACRO INTERSECT

After the initial data step does its "magic", records written out to data set NEWFINISHED are appended to data set FINISHED. Data set FINISHED contains all the CM information that is in its desired form.

If CM data still needs to be processed by algorithm represented above, then do the following:

1) Resort data in arrays
    a. Desired sort order by start date and stop date within start date may not hold true as portions of CMs are processed and output
2) Call macro INTERSECT from inside itself
    a. Passing in the remaining data still requiring the same type of processing
    b. Continue until all data is processed (i.e. "Rinse and Repeat")

Once there are no more CM data to process, all macro calls end *as if by magic*

Save a permanent dataset that contains this different "version" of CMs now stored in data set FINISHED.

RECURSIVE MACRO INTERSECT

Macro intersect is a recursive SAS macro—that is, a SAS macro that can call itself. Here is the code:

```
%macro intersect(data=);
   *** &DATA. IS THE INCOMING DATA SET TO PROCESS - IT WILL CONTAIN ***;
   *** RECORDS IF A CMs FOR A SUBJECT NEED PROCESSING IN A SUBSEQUENT ***;
   *** MACRO CALL ***;
   *** NEWFINISHED CONTAINS RECORDS THAT ARE PROCESSED INTO THE FINAL ***;
   *** DESIRED FORMAT ***;
   data &data.(drop = medcomb astdt aendt)
      newfinished(keep = usubjid trtsdt trtedt medcomb astdt aendt mednum);
      set &data. end=last;
      *** MAXMEDS IS THE MAXIMUM NUMBER OF CMs FOR A SUBJECT ***;
      array med{&maxmeds.} med1-med&maxmeds.;
      array startdt{&maxmeds.} start1-start&maxmeds.;
      array stopdt{&maxmeds.} stop1-stop&maxmeds.;
      *** LENGTH IS LARGE TO ACCOMMODATE CONCATENATION OF MANY CMs ***;
      length medcomb $2000;
      *** SUBJECT ONLY HAS 1 CM – MEDNUM DETERMINE ***;
      *** PRIOR TO MACRO ***;
      if mednum eq 1 then do;
         medcomb=med1;
         *** &NOSTDATE IS A DATE BEFORE ALL STUDY DATA AND IS USED FOR ***;
         *** SIMPLICITY IN ALGORITHM IN HANDLING OF MISSING DATES ***;
         if start1 ne &nostdate. then astdt=start1;
         *** &ONGDATE IS A DATE AFTER ALL STUDY DATA AND IS USED FOR ***;
         *** SIMPLICITY IN ALGORITHM IN HANDLING OF MISSING DATES ***;
         if stop1 ne &ongdate. then aendt=stop1;
         output newfinished;
         mednum=0;
      end;
      else do;
      *** SUBJECT HAS 2+ CMs ***;
          *** KEEP COPIES OF 1ST CM MEDICATION AND ITS STOP DATE ***;
          *** IN CASE NEEDED DUE TO OVERLAPPING OF CMs ***;
         _med1=med1;
         _stop1=stop1;
         *** 1ST CM STARTED BEFORE 2ND CM ***;
         if start1 lt start2 then do;
            *** 1ST CM STOPPED BEFORE 2ND CM STARTED ***;
            if stop1 lt start2 then do;
               *** OUTPUT 1ST CM TO NEWFINISHED ***;
               medcomb=med1;
               astdt=start1;
               aendt=stop1;
               output newfinished;
               *** MOVE ARRAY ELEMENTS TO LEFT – OVERWRITE 1ST ELEMENT ***;
               *** IN EACH ARRAY (CM ALREADY OUTPUT TO NEWFINISHED ***;
               do _i=1 to mednum-1;
                  med[_i]=med[_i+1];
                  startdt[_i]=startdt[_i+1];
                  stopdt[_i]=stopdt[_i+1];
               end ;
               *** DECREMENT NUMBER OF CMs FOR SUBJECT ***;
               mednum=mednum-1;
            end;
            else do;
            *** 1ST CM STOPPED AFTER 2ND CM STARTED ***;
               *** OUTPUT PORTION OF 1ST CM PRIOR TO START OF 2ND CM ***;
               medcomb=med1;
               astdt=start1;
               aendt=start2-1;
               output newfinished;
               *** CREATE COMBINED CM FROM 1ST AND 2ND CMs THAT OVERLAP ***;
               med1=catx(' + ',med1,med2);
               *** START DATE IS START OF 2ND CM ***;
               start1=start2;
               *** DETERMINE STOP DATE OF COMBINE CM ***;
               *** 1ST AND 2ND CMs STOP ON SAME DATE ***;
               if stop1 eq stop2 then do;
                  *** SUBJECT HAS 3+ CMs ***;
               *** MOVE ELEMENTS OF ARRAYS TO LEFT BUT DO NOT ***;
               *** OVERWRITE 1ST ELEMENTS OF ARRAY - CONTAIN NEW ***;
               *** COMBINED CM DATA ***;
               if mednum ge 3 then do;
                  do _i=2 to mednum-1;
                     med[_i]=med[_i+1];
                     startdt[_i]=startdt[_i+1];
                     stopdt[_i]=stopdt[_i+1];
                  end ;
               end;
               *** DECREMENT NUMBER OF MEDICATIONS ***;
               mednum=mednum-1;
            end;
            else if stop1 gt stop2 then do;
            *** 1ST CM STOPS AFTER 2ND CMs STOPS ***;
               *** ASSIGN COMBINE CM STOP DATE OF 2ND CM ***;
               stop1=stop2;
               *** ASSIGN 2ND ARRAY ELEMENT ORIGINAL 1ST CM WHICH ***;
               *** EXTENDS PAST ORIGINAL 2ND CM ***;
               med2=_med1;
               *** ASSIGN 2ND ARRAY ELEMENT DAY AFTER ORIGINAL 2ND ***;
               *** CM STOPPED AS START DATE OF NEW 2ND CM (1ST CM ***;
               *** THAT EXTENDS BEYOND ORIGINAL 2ND CM ***;
               start2=stop2+1;
               *** ASSIGN 2ND ARRAY ELEMENT STOP DATE THE DATE THAT ***;
               *** ORIGINAL 1ST CM STOPPED (PORTION OF 1ST CM THAT ***;
               *** EXTENDS BEYOND ORIGINAL 2ND CM) ***;
               stop2=_stop1;
            end;
            else start2=stop1+1;
            *** 1ST CM STOPS BEFORE 2ND CM STOPS - ASSIGN DAY AFTER ***;
            *** 1ST CM STOPS AS START DATE FOR PORTION OF 2ND CM ***;
            *** EXTENDS BEYOND 1ST CM ***;
         end;
      end;
      else do;
      *** 1ST CM STARTS ON SAME DATE AS 2ND CM STARTS ***;
         *** CONCATENATE 1ST AND 2ND CMs THAT OVERLAP ***;
         med1=catx(' + ',med1,med2);
         *** 1ST AND 2ND CMs STOP ON SAME DATE ***;
         if stop1 eq stop2 then do;
            *** SUBJECT HAS 3+ CMs ***;
            *** MOVE ELEMENTS OF ARRAYS TO LEFT BUT DO NOT ***;
            *** OVERWRITE 1ST ELEMENTS OF ARRAY - CONTAIN NEW ***;
            *** COMBINED CM DATA ***;
            if mednum ge 3 then do;
               do _i=2 to mednum-1;
                  med[_i]=med[_i+1];
                  startdt[_i]=startdt[_i+1];
                  stopdt[_i]=stopdt[_i+1];
               end ;
            end;
            *** DECREMENT NUMBER OF MEDICATIONS ***;
            mednum=mednum-1;
         end;
         else if stop1 gt stop2 then do;
         *** 1ST CM STOPS AFTER 2ND CMs STOPS ***;
```

```
            *** ASSIGN COMBINE CM STOP DATE OF 2ND CM ***;
            stop1=stop2;
            *** ASSIGN 2ND ARRAY ELEMENT ORIGINAL 1ST CM WHICH ***;
            *** EXTENDS PAST ORIGINAL 2ND CM ***;
            med2=_med1;
            *** ASSIGN 2ND ARRAY ELEMENT DAY AFTER ORIGINAL 2ND ***;
            *** CM STOPPED AS START DATE OF NEW 2ND CM (1ST CM ***;
            *** THAT EXTENDS BEYOND ORIGINAL 2ND CM ***;
            start2=stop2+1;
            *** ASSIGN 2ND ARRAY ELEMENT STOP DATE THE DATE THAT ***;
            *** ORIGINAL 1ST CM STOPPED (PORTION OF 1ST CM THAT ***;
            *** EXTENDS BEYOND ORIGINAL 2ND CM) ***;
            stop2=_stop1;
         end;
         else start2=stop1+1;
         *** 1ST CM STOPS BEFORE 2ND CM STOPS - ASSIGN DAY AFTER ***;
         *** 1ST CM STOPS AS START DATE FOR PORTION OF 2ND CM ***;
         *** EXTENDS BEYOND 1ST CM ***;
      end;
   end;
   *** IF ANY REMAINING CMs TO PROCESS, THEN OUTPUT ***;
   if mednum ge 1 then output &data.;
   *** DROP TEMPORARY VARIABLES ***;
   drop _:;
   format astdt aendt date9.;
run ;

data finished;
   *** APPEND PORTION OF DATA THAT IS IN DESIRED FORMAT TO DATA SET ***;
   *** FINISHED ***;
   set finished newfinished(in=innew);
   *** DETERMINE IF ORIGINAL START DATE IS WAS MISSING ***;
   *** IF SO, ASSIGN MISSING (AS A DATE EARLIER THAN STUDY DATA WAS ***;
   *** TO SIMPLIFY ALGORITHM ***;
   if innew then do;
      *** USED TO CALCULATE START TIME FROM TRTSDT - KEEP COPY ***;
      *** IN CASE MISSING START DATE RESTORED FROM ORIGINAL DATA ***;
      _astdt=astdt;
      *** DETERMINE IF ORIGINAL START DATE IS WAS MISSING ***;
      *** IF SO, ASSIGN MISSING (AS A DATE EARLIER THAN STUDY DATA ***;
      *** WAS TO SIMPLIFY ALGORITHM ***;
      if astdt eq &nostdate. then astdt=.;
   end;

run ;
*** MACRO TO DETERMINE NUMBER OF OBSERVATIONS IN DATA SET ***;
%mnumobs(inset=&data.)
*** IF DATA SET HAS 1+ OBSERVATIONS, THEN PROCESS FURTHER AND CALL ***;
*** INTERSECT MACRO AGAIN ***;
%if &dsobs. gt 0 %then %do;

   data &data.;
      set &data.;
      array med{&maxmeds.} med1-med&maxmeds.;
      array startdt{&maxmeds.} start1-start&maxmeds.;
      array stopdt{&maxmeds.} stop1-stop&maxmeds.;
      *** SORT REMAINING DATA - AS PROCESS ABOVE MAY NOT RESULT IN ***;
```

```
         *** EXPECTED ORDER FOR ENTRY INTO MACRO ***;

         *** SORT BY START DATE ***;
         do until(sorted);
            sorted=1;
            do _i=1 to mednum-1;
               if startdt[_i] gt startdt[_i+1] then do;
                  _med=med[_i+1];
                  med[_i+1]=med[_i];
                  med[_i]=_med;
                  _startdt=startdt[_i+1];
                  startdt[_i+1]=startdt[_i];
                  startdt[_i]=_startdt;
                  _stopdt=stopdt[_i+1];
                  stopdt[_i+1]=stopdt[_i];
                  stopdt[_i]=_stopdt;
                  sorted=0;
               end;
            end ;
         end ;

         *** SORT BY STOP DATE WITHIN START DATE ***;
         do until(sorted);
            sorted=1;
            do _i=1 to mednum-1;
               if startdt[_i] eq startdt[_i+1] then do;
                  if stopdt[_i] gt stopdt[_i+1] then do;
                     _med=med[_i+1];
                     med[_i+1]=med[_i];
                     med[_i]=_med;
                     _startdt=startdt[_i+1];
                     startdt[_i+1]=startdt[_i];
                     startdt[_i]=_startdt;
                     _stopdt=stopdt[_i+1];
                     stopdt[_i+1]=stopdt[_i];
                     stopdt[_i]=_stopdt;
                     sorted=0;
                  end;
               end;
            end ;
         end ;
         *** DROP TEMPORARY VARIABLES ***;
         drop _:;
      run ;
      *** RECURSIVE CALL OF INTERSECT MACRO ON REMAINING DATE TO BE ***;
      *** PROCESSED ***;
      %intersect(data=&data.)
   %end;
%mend;
*** ORIGINAL CALL OF MACRO INTERSECT ***;
%intersect(data=adcm4)
```

COMBINE WITH LAB DATA

The final CM data set contains records of what CMs a subject is taking at any given time. This new combination CM data set can be easily combined with lab data to produce the output displayed earlier

(**Listing 1. Example Listing of CMs Taken in Relation to Laboratory** Data and **Table 1. Example Table of CMs Taken in Relation to Laboratory** Data).

The code to accomplish this is:

```
proc sql noprint;
   create table combine(where=(aftastdt)) as
   select s.*, l.paramcd, l.adt, l.aval, l.anruln,
    s.astdt lt l.adt and (s.aendt eq . or l.adt le s.aendt) as aftastdt
   from adstatds as s left join adlb as l
   on s.usubjid eq l.usubjid
   order by s.usubjid <,DESIRED VARIABLES>;
quit;
```

CONCLUSION

A recursive macro (intersect) processes CM data of an unknown and potentially changing complexity resulting in data that represent all CMs being taken by a subject at any given time. Macro intersect simplifies this process by:

- Only looking at the first two CMs to determine
  - What can be output at that point
  - Preparing data for the same, simple processing in a recursive macro call
  - Eliminating the need to have super complex code to handle complexity and future changes in data.

Eating the elephant one bite at a time!

ACKNOWLEDGMENTS

Ben Young – Boston Scientific

Indu Nair, Katy White – United BioSource LLC

CONTACT INFORMATION

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

Scott Horton
popcornasaurus@gmail.com