Merging Multiple Observation Data Sets Together
Tim Kelly, Synteract, Inc, Ambler, PA

Abstract
The review of clinical data can sometimes call upon a programmer to be very creative in order to conquer complex issues. These desired results can cause angst and frustration in getting the objective done. One such nuisance is to generate a comfortable output from multiple data sets, where more than one of these data sets contains multiple observations. When these data sets cannot be merged together without causing a “many to many merge”, and Proc SQL code is not generating the desired result, this paper will give a simple example on how to conquer this complex issue.

Clean Merge
Every programmer needs to overcome the tendency to overcomplicate an issue. Sometimes we are so blinded with producing a specific result that we are unable to see the path of how to get there. There are many obstacles to overcome during the creation of a program. One such hurdle is merging data in which the uniqueness of each data set involved does not allow a “clean” merge. The following is an example of a “clean” merge. For simplicity, I’ll use the merging of only two data sets. The uniqueness of the data sets, per the BY variables, i.e. PATID, has the following attributes:

<table>
<thead>
<tr>
<th>Data Set A</th>
<th>Data Set B</th>
</tr>
</thead>
<tbody>
<tr>
<td>One observation</td>
<td>One observation</td>
</tr>
<tr>
<td>One observation</td>
<td>Many observations</td>
</tr>
<tr>
<td>Many observations</td>
<td>One observation</td>
</tr>
</tbody>
</table>

Then we will have a “clean” merge. On the contrary, if we have many observations in both Data Set A and Data Set B, we will get a message in the Log after the merge which would state: ‘MERGE statement has more than one data set with repeats of BY values.’ This message indicates that we do not have a clean merge, and we would need to modify the BY variables in order to ensure the merge is happening appropriately.

The Problem
But, what if modifying the variables on the BY statement does not solve the problem of the many to many merge? What if currently, there are not any variables between the data sets, which would allow the data to merge “cleanly”? What if the use of Proc SQL is not able to fix the problem? Now that the problem has become complicated, we want to make sure we don’t overcomplicate the solution.

The Task
The assignment given to the programmer is to put together a listing for the data manager. This listing would comprise of specific information between the patient’s Concomitant Medication and Adverse Event data. Each patient may have many adverse events, as well as multiple medications they are taking during the study. The sole commonality between the two files is the patient ID number. The data manager requests that the output be displayed horizontally, because the number of patients in the database is quite extensive. We want to minimize the output to save paper, and we want to make it easy for the data manager to review. Given the above information, we are going to have a problem merging the data together. Since the only commonality in the current data sets is the patient ID number, which we know will occur many times in each data set, we are going to create a new variable to help make a “clean” merge.

Programming Example
The following coding example will create a new variable called COUNT. This variable will be a unique identifier that is simply the observation number, per patient, in each data set. We will then use the patient ID number and the patient’s newly created observation number to merge the two data sets together. The newly created variable COUNT is just a temporary variable that is used solely for the purpose of the listing, and will not exist permanently in the database.
/* Sort Adverse Event data set, keeping only the necessary variables, so we can count the number of events (observations) each patient has */

proc sort data = db.ae
  out = ae(keep= patid aename dtaebeg aesev);
  by patid aename dtaebeg;
  where (aename ne ' ');
run;

/* count the number of obs in AE per patient */
data aecnct;
  set ae;
  by patid aename dtaebeg;
  retain count 0;
  if (first.patid) then count=1;
  else count+1;
run;

/* Sort Concomitant med data set, keeping only the necessary variables, so we can count the number of meds (observations) each patient has */

proc sort data = db.conmeds
  out = conmeds(keep=patid cmname dtcmbg indicate);
  by patid cmname dtcmbg;
  where (cmname ne ' ');
run;

/* Count the number of obs in Conmed per patient */
data cmcnt;
  set conmeds;
  by patid cmname dtcmbg;
  retain count 0;
  if (first.patid) then count=1;
  else count+1;
run;

/* merge AE and CONMEDS by patient observation number keeping all observations */
data final(drop=count);
  merge aecnct (in=inae)
    cmcnt (in=incm);
  by patid count;
run;

proc print data = final;
run;

Summary

The final data set will contain both the Adverse Event and Concomitant Medication information displayed horizontally. The number of observations in the new FINAL data set, per patient, will be the maximum number of observations that patient has in either the Adverse or Conmed data sets. The output will have the minimum number of observations as possible per patient, while displaying the data together on the same line. Note that the Adverse Event and Conmed data is NOT directly related, even though it would appear on the same line.

The advantage to programming this way is to save space, especially when you have an enormous amount of data. It also prevents duplicate information from showing up in your output, which a Proc SQL would do in this case.

One of the greatest benefits to programming is the independent thought and creativity it allows. A desired result can be attained by a number of different methods. This is just one approach that simplifies a complex problem without overcomplicating it.

REFERENCES

SAS® is a registered trademark of the SAS Institute Inc., Cary, NC USA.

ACKNOWLEDGEMENTS

I’d like to thank Mike Rea for his helpful feedback and editing. I’d also like to thank Daphne for her programming knowledge, which she has passed along to me.

AUTHOR INFORMATION

The author of this paper can be contacted as follows:

Tim Kelly, Synteract, Inc.
Voice: (215) 283-9370 x608
FAX: (215) 283-9366
EMS: tkelly@synteract.com