PharmaSUG 2018 - Paper BB-23

Beyond MERGE for Combining Datasets

David Franklin, IQVIA, Cambridge, MA

ABSTRACT

Merging two datasets is one of the most common actions that a SAS® programmer does when manipulating data to bring it into a form for either storage or analysis. The most common way data is merged is with the use of the MERGE statement inside a DATA step but there are others. This paper will introduce methods that use PROC SQL, HASH tables, POINT and KEY options in a SET statement and PROC FORMAT; looking at one-to-one, one-to-many and many-to-many situations.

INTRODUCTION

Merging data from two datasets and putting the result into a third dataset is one of the basic data manipulation tasks that a SAS programmer has to do. Before we introduce the methods, let's look at some data:

```
Dataset: PATDATA

SUBJECT  TRT_CODE
124263      A
124264      A
124265      B
124266      B

Dataset: ADVERSE

SUBJECT  EVENT
124263   HEADACHE
124266   FEVER
124266   NAUSEA
124267   FRACTURE
```

This data has a one-to-many structure and will be used in this section of the paper. The same methods apply using a one-to-one data structure.

The most common way two datasets are joined is using the MERGE statement within a DATA step, as shown below:

```
DATA _alldat0;
  MERGE adverse (in=a)
        patdata (in=b);
  BY subject;
  IF a;
RUN;
```

The dataset that is produced is:

```
SUBJECT  EVENT     TRT_CODE
124263   HEADACHE     A
124266   FEVER        B
124266   NAUSEA       B
124267   FRACTURE
```

WIth the use of the 'IF a;' in the DATA step, an effective left join of the data is completed resulting in all records from dataset ADVERSE being in the outgoing dataset, and only those records from PATDATA being a match, included.

OTHER METHODS

There are a number of other ways the data can be merged, the next common using a PROC SQL as shown below:

```
PROC SQL;
  CREATE TABLE _alldat0 AS
   SELECT a.*, b.trt_code
   FROM adverse a LEFT JOIN patdata b
   ON a.subject=b.subject;
  QUIT;
RUN;
```

SQL is a well known language that is very good at working with databases and is liked by many who deal with large datasets and has the advantage of not sometimes needing a PROC SORT call for both datasets before the MERGE statement.

Creating a format from the PATDATA dataset is another way that data can be merged, as shown below:

```
DATA fmt;
 RETAIN fmtname 'TRT_FMT' type 'C';
 SET patdata;
 RENAME subject=start trt_code=label;
PROC FORMAT CNTLIN=fmt;
DATA alldata0;
 SET adverse;
 ATTRIB trt_code LENGTH=$1 LABEL='Treatment Code';
 trt_code=PUT(subject,$trt_fmt.);
RUN;
```

In the example a character format TRT_FMT is created from the PATDATA dataset, and then this format is used to set the TRT_CODE variable within the ADVERSE dataset. Like the PROC SQL, the datasets do not need to be merged.

The SET statement has the KEY= option which is useful for merging as shown in the following example:

```
DATA _alldat0;
 SET adverse;
 SET patdata KEY=subject /UNIQUE;
 DO;
  IF _IORC_ THEN DO;
   _ERROR_=0;
   trt_code='';
  END;
 END;
RUN;
```

Before this type of merge can work the dataset PATDATA must have an index created inside it, using either the INDEX statement inside a DATASETS or SQL procedure, or INDEX option inside a DATA step. It is important to have the DO loop is if no match is found then TRT_CODE will be set to missing - if this is not done then unexpected results may occur.

The Hash Table is used by database programmers in other languages, this is considered one of the fastest ways to merge data in two datasets. Many papers have been written about this recent feature, how it works, and their use within SAS - references to some notable papers are the Reference section below. The code below does the merge required:

```
DATA _alldat0;
 IF _n_=0 THEN SET patdata;
 IF _n_=1 THEN DO;
  DECLARE HASH _h1 (dataset: "PATDATA");
  rc=_h1.definekey("SUBJECT");
  rc=_h1.definedata("TRT_CODE");
  rc=_h1.definedone();
  call missing(SUBJECT,TRT_CODE);
 END;
 SET adverse;
 rc=_h1.find();
 IF rc^=0 THEN trt_code=" ";
 DROP rc;
RUN;
```

In the example above, the dataset PATDATA gets loaded into a hash table, then the ADVERSE dataset is loaded into the DATA step and the match is made using the FIND() method.

MERGING A MANY-TO-MANY DATA STRUCTURE

When two datasets to join have no unique record structure this is called a many-to-many merge. To do this correctly there are a couple of ways that this can be done, but first some data:

```
DATASET: ADVERSE

SUBJECT  DATE       EVENT
342001   16NOV2017  Nausea
342002   16NOV2017  Heartburn
342002   16NOV2017  Acid Indigestion
342002   18NOV2017  Nausea
342003   17NOV2017  Fever
342003   18NOV2017  Fever
342005   17NOV2017  Fever

DATASET: CONMED

SUBJECT  DATE       DRUG
342001   16NOV2017  Dopamine
342002   16NOV2017  Antacid
342002   16NOV2017  Sodium bicarbonate
342002   18NOV2017  Dopamine
342003   18NOV2017  Asprin
342004   19NOV2017  Asprin
342005   17NOV2017  Asprin
```

Using PROC SQL, which is the most common method for this structure, the code would be written as:

```
PROC SQL;
 CREATE TABLE _alldat0 AS
 SELECT a.*, b.drug
 FROM adverse a INNER JOIN conmed b
 ON a.subject=b.subject AND a.date=b.date;
 QUIT;
RUN;
```

A call to output the list to the Results window with PROC SQL would produce:

```
subject      date  event               drug
-----------------------------------------------------
342001      21139  Nausea              Dopamine
342002      21139  Heartburn           Antacid
342002      21139  Heartburn           Sodium bicarbonate
342002      21139  Acid Indigestion    Antacid
342002      21139  Acid Indigestion    Sodium bicarbonate
342002      21141  Nausea              Dopamine
342003      21141  Fever               Asprin
342005      21140  Fever               Asprin
```

Another approach is to merge the data of this form is using a loop within a loop and the POINT option in the SET statement:

```
DATA _alldat0;
 SET adverse;
 DROP _:; ** Drop temporary variables;
 match=0; ** Match flag;
 ** Our loop within a loop -- output if match;
 DO i=1 TO xnobs;
  ** Need to rename the "merging" variables within the CM dataset;
  SET conmed (RENAME=(subject=_subject date=_date)) NOBS=xnobs POINT=i;
   ** Have to rename matching variables so that they do not overwrite
      the original values in AE;
  IF (subject=_subject AND date=_date) or (subject=_subject AND date>. and _date=.)
THEN DO;
   match=1; ** Yes, there is a match my the "merging" variables;
   OUTPUT;
  END;
 END;
RUN;
```

This is a lot of code but does give the most control and the same results. An important note here is that the matching variables have to be renamed so that they do not overwrite the original values in ADVERSE (very important) but use the DROP statement to get rid of these when the dataset _ALLDAT0 is created. Note also that the variable MATCH is used so that it is easy to see where a match is made but is not necessary.

CONCLUSION

As can been seen there are a number of methods which can be used to merge data, beyond the MERGE statement within a DATA step. No one method is better than another and does depend on a number of factors including size of data and whether it is sorted or indexed first. Also, the methods shown here are by no means exhaustive and it is only though trying these different methods at your installation that you will see resource efficiencies between the methods.

CONTACT INFORMATION

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

David Franklin
IQVIA
david.franklin1@iqvia.com