Beyond the Simple SAS Merge

Vanessa L. Cox, MS[1,2], and Kimberly A. Wildes, DrPH, MA, LPC, NCC[3]

[1]General Internal Medicine and Ambulatory Treatment, The University of Texas MD Anderson Cancer Center, Houston, TX. vlcox@mdanderson.org

[2]PhD Candidate, Department of Statistics, Texas A&M University

[3]Private Practice, Houston, TX. info@kimberlycounseling.com

Background

SAS is a powerful data management tool. Used correctly, its merge statements can ensure accuracy in combining datasets. The most basic merge types are one-to-one and one-to-many, both of which only require each data set (1) to contain one common identifying variable and (2) to be sorted in the same order by the identification (id) variable.

A one-to-one merge is the merging of two datasets, each of which has only one record per subject. This merging essentially adds the variables from one dataset to an original or pre-existing dataset. Suppose you have *data1*, a dataset containing case_id and age, and *data2*, a dataset containing case_id and gender. After a one-to-one merge you will have the dataset *data3*, which will contain case_id, age, and gender.

In the case of a one-to-many merge, one dataset still has one observation per case. The other dataset, however, contains the observation ids with more than one record per unique id. Such would be the case if data were collected at multiple visits. Thus, *data1* might contain case_id and demographics, and *data2* contain case_id and several records per patient, for example, one for each time blood pressure was recorded.

Beyond these simple merge scenarios, though, there may be cases when a many-to-many merge is needed. Suppose a researcher has two sets of data: one has patients with multiple visits (1-5 visits), and the other has patients with multiple drugs prescribed (1-5 meds). Trying to sort the first dataset by case_id and visit, sorting the second by case_id and drug, and then trying to merge them, results in a warning in the SAS log that the merge statement contains more than one set of by variables. Since there are times when a many-to-many merge is necessary, and a simple SAS merge statement would not accomplish this action, the aim of this project was to implement use of proc sql to accomplish this more complex merge and then to compare the method to other widely used merging algorithms.

Methods

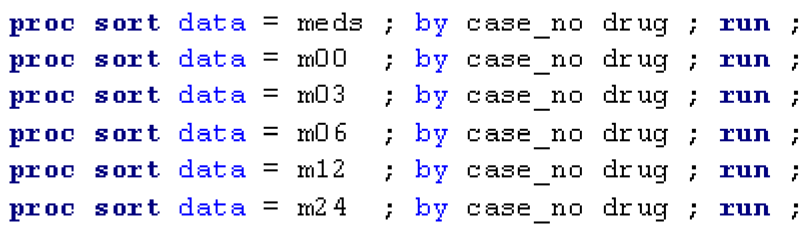
One way that a many-to-many merge can be accomplished is by a series of one-to-many merges. The dataset needs to be sorted by a combination of variables that lead to unique observations. For example, one might sort by case_id and visit number. After this one-to-many merge, there would be a dataset with as many rows as the dataset with the multiple visits, and each record would contain the demographic data. Special care can be taken to retain observations that appear in only one dataset. See Example 1 below:

Example 1: Procedure to merge datasets using set statements:

- Step 1: Divide visits dataset into multiple datasets, each having unique case id's.

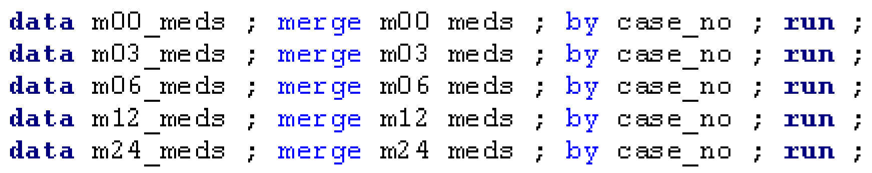
```
data m00 ; set visits ; if visit_no = 'V00' ;
data m03 ; set visits ; if visit_no = 'V03' ;
data m06 ; set visits ; if visit_no = 'V06' ;
data m12 ; set visits ; if visit_no = 'V12' ;
data m24 ; set visits ; if visit_no = 'V24' ;
```

- Step 2: Sort the broken up visits and the medication dataset by the same variable so they can be merged.



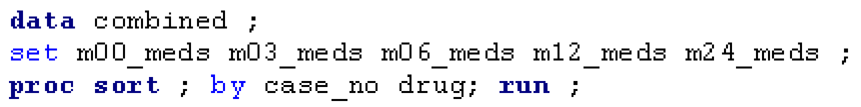
```
proc sort data = meds ; by case_no drug ; run ;
proc sort data = m00  ; by case_no drug ; run ;
proc sort data = m03  ; by case_no drug ; run ;
proc sort data = m06  ; by case_no drug ; run ;
proc sort data = m12  ; by case_no drug ; run ;
proc sort data = m24  ; by case_no drug ; run ;
```

- Step 3: Merge each visit dataset to the medications database.



```
data m00_meds ; merge m00 meds ; by case_no ; run ;
data m03_meds ; merge m03 meds ; by case_no ; run ;
data m06_meds ; merge m06 meds ; by case_no ; run ;
data m12_meds ; merge m12 meds ; by case_no ; run ;
data m24_meds ; merge m24 meds ; by case_no ; run ;
```

- Step 4: Combine them all by a set statement.



```
data combined ;
set m00_meds m03_meds m06_meds m12_meds m24_meds ;
proc sort ; by case_no drug; run ;
```

Merging in this way would work if there were only a few visits, or a few medications, but the merge process could easily get much more complicated if the number of variables to sort by increased. As an alternative, a few lines of code using proc sql would accomplish the same many-to-many merge. See Example 2 below for sample code, which functions similar to the code for joining two tables in Microsoft Access.

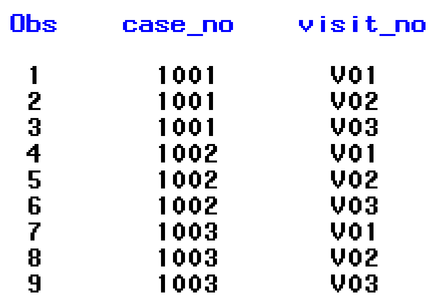
Example 2: Proc sql code used for this merge:

```
proc sql ;
create table new_sql as *
select visits.*, drugs.*
from visits, drugs
where visits.case_id = drugs.case_id; quit ;
```

Results

SAS proc sql code is more concise and leads to clean, straight-forward datasets. See the sample datasets (*data1*, *data2*, *data3*) below. *Data1* contains patient ids and multiple visit numbers per patient. *Data2* contains patient ids and multiple drugs per patient. *Data3* is the resulting dataset from a many-to-many merge using proc sql.

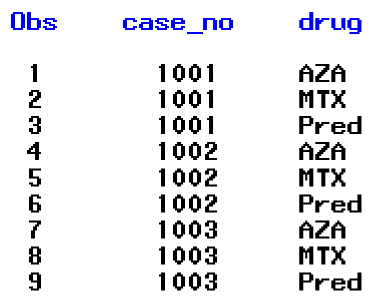
Sample data1 - Original Visits dataset:



```
Obs    case_no    visit_no

 1      1001        V01
 2      1001        V02
 3      1001        V03
 4      1002        V01
 5      1002        V02
 6      1002        V03
 7      1003        V01
 8      1003        V02
 9      1003        V03
```

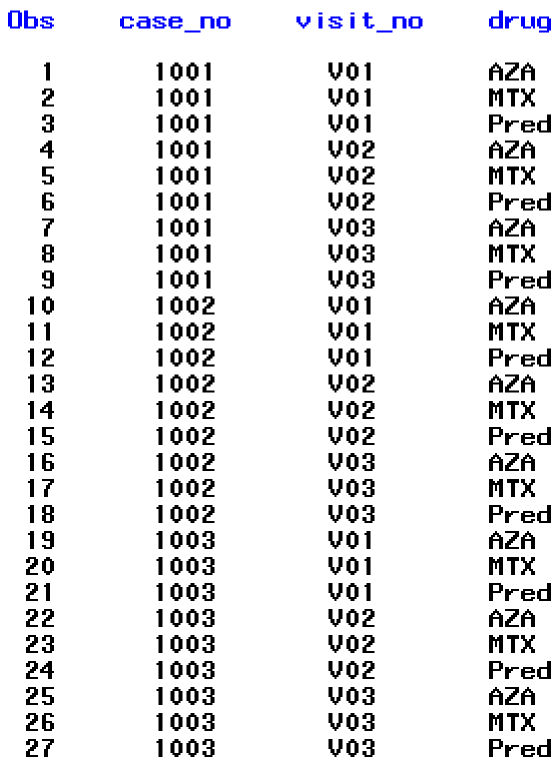
Sample data2 - Original Drugs dataset:



```
Obs    case_no    drug

 1      1001      AZA
 2      1001      MTX
 3      1001      Pred
 4      1002      AZA
 5      1002      MTX
 6      1002      Pred
 7      1003      AZA
 8      1003      MTX
 9      1003      Pred
```

Sample data3 - Merged visits and drugs from proc sql:



```
Obs    case_no    visit_no    drug

  1     1001        V01       AZA
  2     1001        V01       MTX
  3     1001        V01       Pred
  4     1001        V02       AZA
  5     1001        V02       MTX
  6     1001        V02       Pred
  7     1001        V03       AZA
  8     1001        V03       MTX
  9     1001        V03       Pred
 10     1002        V01       AZA
 11     1002        V01       MTX
 12     1002        V01       Pred
 13     1002        V02       AZA
 14     1002        V02       MTX
 15     1002        V02       Pred
 16     1002        V03       AZA
 17     1002        V03       MTX
 18     1002        V03       Pred
 19     1003        V01       AZA
 20     1003        V01       MTX
 21     1003        V01       Pred
 22     1003        V02       AZA
 23     1003        V02       MTX
 24     1003        V02       Pred
 25     1003        V03       AZA
 26     1003        V03       MTX
 27     1003        V03       Pred
```

Conclusion

A short proc sql statement can accurately merge all variables across two datasets, without the necessity of paying too close of attention to the possible values of any variables. SAS will merge for each case at every level present. For small datasets with only a few combinations, merging is fairly easy to accomplish using merge statements by: 1) creating a series of selections by a variable, 2) merging with the unique combinations and , 3) appending all datasets together using a set statement. However, as this project concludes, proc sql efficiently accomplished the same merge for any combination of dataset types needed. Furthermore, using proc sql might also prevent coding errors derived from copying and pasting or other common programming or data manipulation errors.