Paper CC-04

Accuracy or Efficiency in Merging

Cindy Liu, Wells Fargo Bank, West Des Moines, IA
Dongmei Lan, Cincinnati Children's Hospital, Cincinnati, OH

ABSTRACT

Merge is an often-used statement in SAS programming. Besides one-to-one merging and match-merging, sometimes we need merge datasets with specific conditions. For instance, we want to change the value of variable after merging. These tasks normally involve IF-THEN or SELECT statement. The efficient solution is to apply MERGE, IF-THEN or SELECT statements in one data step. The results would be inaccurate in some situation.

In this paper, several error examples are given to show the outcomes when we apply the seemly efficient method described above. Specific care has to be taken when trying to find an efficient algorithm for data merging without losing accuracy.

INTRODUCTION

SAS provides several ways to combine datasets together. The SET and MERGE statements are two main tools, which are flexible and have a variety of uses in SAS programming, especially in the one-to-multiple combination. SET is commonly used to combine one observation with many while MERGE performs the one-to-multiple merge with a specific condition, i.e. to combine the observation according to the values of common variables, which is also called Match-Merging.

In the real application, along with the combination, it is not unusual to modify values of some variables for observations that satisfy specific conditions. IF-THEN and SELECT are often used to perform this task.

When we perform the combination and modification in the same data step, results may not be correct. It has been found that there are unexpected results when MERGE and IF-THEN statements are used within one data step (Yan Lei 1). Same type of error happens when use SET and IF-THEN or MERGE and SELECT. Several examples are presented below to show the error.

EXAMPLE I: MERGE + IF-THEN

There are two datasets one and two, each of them are shown as follows:

```
Data one
x y
1 2
2 3
4 5
```

```
Data two
x z
1 4
1 7
1 10
2 8
4 9
```

The following code merges the two datasets and changes the value of y to 7 when z equals to 7.

```
data three;
    merge one two;
    by x;
    if z=7 then y=7;
run;

proc print data=three;
run;
```

The execution result from above code is as follows:

Example I output

The SAS System

Obs	x	y	z
1	1	2	4
2	1	7	7
3	1	7	10
4	2	3	8
5	4	5	9

We want to change the value of y only when z=7. However in above example, in observation 3, z equals to 10 and the y supposes to be 2 instead of 7. This is because the value of y is inherited from observation 2 as '7', although the 'if 'condition is not satisfied in observation 3 for z value and 'then' statement is unexecuted. The iteration joined z value to dataset one and kept y value from previous observation. This is not correct.

EXAMPLE II: MERGE + SELECT

Similar to the MERGE + IF-THEN example above, this example describes a situation using MERGE and SELECT statement with one data step to perform conditional processing.

```
data four;
    merge one two;
    by x;
    select (z);
           when (7) y=7;
           otherwise ;
    end;
run;

proc print data=four;
run;
```

The execution result from above code is as follows:

Example II output;

The SAS System

Obs	x	y	z
1	1	2	4
2	1	7	7
3	1	7	10
4	2	3	8
5	4	5	9

Again, in observation 3, the value of y is not the expected value for z=10.

EXAMPLE III: SET + IF-THEN

The similar situation exists when we use multiple SET statements to do one-to-multiple combination reading in a DATA step.

There are two datasets six and two, each of them are shown as follows:

```
Data six
u v
2 8
```

```
Data two
x z
1 4
1 7
1 10
2 8
4 9
```

The following code reads datasets six and two into dataset seven, modifies the value of u when z equals to 10 as well.

```
data seven;
    if _n_=1 then set six;
           set two;
    if z=10 then u=3;
run;

proc print data=seven;
run;
```

Example III output;

The SAS System

Obs	u	v	x	z
1	2	8	1	4
2	2	8	1	7
3	3	8	1	10
4	3	8	2	8
5	3	8	4	9

The execution result from above code is as follows:

The value of u should be changed only when z=10 as shown in observation 3. However, the execution returns the result that the values of u in observation 4 and 5 are also changed. These values are incorrect for these two observations.

REASON

We notice that in previous examples, after the value is changed, it remains the same for the subsequent observations. It is known that in the one-to-multiple combination, SAS retains the values of all variables in the program data vector except those variables that are created by the data step. For each BY group, the information from dataset with the less number of observations will not change until SAS finishes processing all observations in the dataset with largest number of observations. For example, in the example III, the u and v values are read into the program data vector only once in the whole data step. In the example I and II, if there is no modification, x=1 and y=2 should remain in the program data vector for the first 3 observations in data three.

In addition, IF-THEN and SELECT change the value in the program data vector. SAS will retain the modified values until there is a change in the data step. It means if there is no change in the data step, the value will retain for the subsequent observations for combination or Match-Merging.

SOLUTION

Since SAS retains the value for variables that are not created by the data step during the one-to-multiple combination, there are two ways to solve this problem. One solution is to merge the data in one data step then modify the data in a second data step. Another one is to generate a new variable which will be updated for each observation in the program data vector.

REMEDY FOR EXAMPLE I AND II (METHOD 1)

In order to fix the problem and get the correct result, two data steps are used. In the first data step, dataset one and two are merged to create dataset five. Then we modify the value of y in the next data step.

```
data five;
    merge one two;
    by x;
run;

data five;
    set five;
    if z=7 then y=7;
run;

proc print data=five;
run;
```

The execution result from above code is as follows:

Example I,II Remedy Output

The SAS System

Obs	x	y	z
1	1	2	4
2	1	7	7
3	1	2	10
4	2	3	8
5	4	5	9

The problem is solved. The value of Y is changed only when Z =7.

REMEDY FOR EXAMPLE III (ALTERNATIVES 1 & 2)

We have two alternatives to fix the problem in example III.
Alternative 1, we have to give up the short-cut idea and go back to what we normally do. We have to use a SET statement to fulfill one-to-one reading in a data step and an IF-THEN statement in the subsequent data step to change the value of u once the condition is met.

```
data eight;
    if _n_=1 then set six;
    set two;
run;

data eight;
    set eight;
    if z=10 then u=3;
run;

proc print data=eight;
run;
```

The execution result from above code is as follows:

Example III Remedy Alternative 1 Output

The SAS System

Obs	u	V	x	z
1	2	8	1	4
2	2	8	1	7
3	3	8	1	10
4	2	8	2	8
5	2	8	4	9

Now we get the result we expect, change of the value of u happens only in observation 3 when z=10.

Alternative 2, the new variable k is generated in the same data step. The values of k equal to the values of u except when z=10. Since k is new variable in this data step, the value of k in program data vector is updated for each observation. SAS does not retain the value of k.

```
data nine;
    if_n_=1 then set six;
    set two;
    if z=10 then k=3;
```

```
            else k=u;
   run;

   proc print data=nine;
   run;
```

The execution result from above code is as follows:

Example III Remedy Alternative 2 Output

The SAS System

obs	u	V	x	z	k
1	2	8	1	4	2
2	2	8	1	7	2
3	2	8	1	10	3
4	2	8	2	8	2
5	2	8	4	9	2

Also we get the result we expect, change of the value of k happens only in observation 3 when z=10. We can use the value of k rather than u.

Comparing the two alternatives, alternative 1 is more practical.

CONCLUSION

When we combine data sets and modify existing variables, we should not try to put MERGE and IF-THEN or MERGE and SELECT in one data step. For the same reason we should not have IF-THEN statement nested within multiple SET statements either. Bear in mind, errors may incidentally occur when we attempt these short cuts. To avoid possible merging errors, the standard and reliable method is to first merge data using MERGE statement or read data using SET statement in one data step and then apply IF-THEN or SELECT statement in next data step. An alternative solution is to modify a new variable which is generated from the existing one. An extra data step or an extra variable seems inefficient. However, it is necessary to guarantee the accuracy of the data in this situation.

REFERENCES

[1] Yan Lei, Be Careful When You Merge SAS Datasets, PharmaSUG 2004
[2] SAS Online Documentation, Version 8

ACKNOWLEDGMENTS

The authors would like to thank Bin Huang, Mekibib Altaye and Robert Tamer, Cincinnati Children's Hospital, Juntao Xu, Bioforce NanoScience for their review and valuable suggestions of this paper.

CONTACT INFORMATION

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

Cindy Liu
Wellsfargo Bank
7000 Vista Drive
West Des Moines, IA 50266
Work Phone: (515)222-8659
Fax: (515)222-8810
E-mail: cindy.liu@wellsfargo.com

Dongmei Lan
Cincinnati Children's Hospital Medical Center
ML 5041
3333 Burnet Ave
Cincinnati, OH 45220
Work Phone: (513)636-9762
Fax: (513)636 -1254
E-mail: dongmei.lan@cchmc.org