SAS Tutorial: Set Merge and Update

Peter L. Rikard

Introduction

This tutorial was presented as a set of examples to demonstrate the combining of datasets in SAS. In the interests of trees and cost of the Proceedings, the examples are not printed here. Instead, at the end of the text is the SAS code to produce all of the examples. An example consists of the print of each dataset used to build the output dataset and the print of the result.

Combining Datasets

A major power of SAS over distant rivals lies in its ability to manipulate data. Taking in information in many forms, in different order, and in different quantity and being able to put it together in a coherent form EASILY, astounds users of that other package, not to be named here. The three tools for this process; SET, MERGE and UPDATE will only be introduced. The possible combinations of things that can be done are huge.

Enough puffery.

Definitions

SAS operates on "rectangular data arrays". Admittedly jargon, but a very simple concept. It is a normal table of data.

<table>
<thead>
<tr>
<th>NAME</th>
<th>SEX</th>
<th>RACE</th>
<th>AGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Peter</td>
<td>1</td>
<td>W</td>
<td>13</td>
</tr>
<tr>
<td>Frank</td>
<td>1</td>
<td>M</td>
<td>11</td>
</tr>
<tr>
<td>Jane</td>
<td>2</td>
<td>W</td>
<td>14</td>
</tr>
<tr>
<td>Susan</td>
<td>2</td>
<td>W</td>
<td>12</td>
</tr>
<tr>
<td>Trinity</td>
<td>3</td>
<td>W</td>
<td>12</td>
</tr>
<tr>
<td>Adele</td>
<td>2</td>
<td>B</td>
<td>10</td>
</tr>
<tr>
<td>Bob</td>
<td>1</td>
<td>W</td>
<td>18</td>
</tr>
</tbody>
</table>

The columns of the table are VARIABLES, things that can take on one or more values, as SEX could have 3 values 1=male, 2=female, 3=unknown. A row of the table is an OBSERVATION, the collection of all the values of variables pertaining to a single entity, a persons NAME, AGE, SEX, HEIGHT, WEIGHT etc. The whole table is a dataset, a rectangular data array, the collection of all the observations.

SAS can read in many types of data in many forms. Reading in data from non SAS datasets is done with INPUT statements and transforms the "raw" data into a SAS dataset. Once this process has been done SAS handles all the chores of WHERE the data is and WHAT does it look like. You may now treat the SAS dataset as a rectangular data array, no matter what its "raw" form looked like, and no matter the form it was stored in. Further use of the SAS datasets is done with SET, MERGE and UPDATE statements, you do NOT use INPUT statements in DATA steps using these previously created datasets.

Combining datasets can be described in many ways, however the names that are used here have particular meanings. Concatenating datasets together so that we get a new dataset that contains the sum of the observations of the input datasets is generally done with a SET statement. If we had individual datasets for each class in a school containing one observation for each person in the class and combined all those datasets with a SET statement, we would have a LONGER dataset containing one observation for each person in EACH class in the school.

```
DATA TOTAL;
SET CLASS1 CLASS2 ...;
```
Merge is a term used in many ways, but in SAS it means one thing in particular. MERGEing two (or more) datasets in SAS creates a dataset where an observation in one dataset is joined to the end of a corresponding observation in another dataset, creating an observation with all the variables in both datasets. (making the table WIDER).

DEMOGRAPHICS

TEST SCORES

DATA TOTAL;
MERGE DEMOGRAP TESTS,
TOTAL

UPDATE is a special form of the merge. It combines the NON-missing values of variables from one dataset with an observation in a master dataset (updates).

The DATA VECTOR. SAS as it is processing each observation in a DATA step, keeps track of all the current values of all the variables "mentioned" (used) in the DATA step. The variables are in the data vector in the order that they are first mentioned, and how SAS handles them is dependent on where they came from and whether they are RETAINED or not.

In general, variables created in the data step are set to missing before SAS starts processing any statements, FOR EACH observation. To prevent this, to keep values in variables from one observation to the next, you use a RETAIN statement, listing the names of the variables that you do NOT want SAS to automatically set to missing. When SAS executes a SET statement, it sets all variables coming from the SAS datasets in the SET statement in the data vector to missing and then moves the values from the dataset into the data vector. This operation is different in the case of the MERGE and UPDATE statements.

To describe the operations I created three datasets, DATA1, DATA2, and DATA3. Not very imaginative names but sufficient. (see source code)

The SET Statement

The most frequently used of the three methods of referencing to SAS datasets is the SET statement. A SET statement may refer to from 1 to 50 SAS datasets at a time. A simple use is to build a new dataset from a single old dataset, either to create new variables, get rid of old ones or select observations. (FIRST EXAMPLE) Here we have created a new dataset from DATA1 by selecting only those observations where the value of one of the variables; J, is greater than 5 and less than 9. The new dataset has all of the variables from DATA1, but only three observations.

Combining datasets. The second example combines the two datasets mentioned in the SET statement, DATA1 and DATA3. The new dataset has all the observations from both AND all the variables from both. The variables Y and Z from DATA3 have been added to the dataset. Since they were not originally in DATA1, when SAS set all variables in the data vector to missing before it built each observation, that is the value that they have in the output dataset (missing). SAS built the data vector, first using variables from DATA1 and then DATA3. Even though the variables were in different order in the two datasets, SAS puts them together correctly. SAS takes one observation at a time from the first dataset mentioned in the SET statement, performs any processing statements and when all observations have been used from the first dataset, it takes observations from the next dataset mentioned in the SET statement ....

The next example (EXAMPLE 3) uses the same statements as the last, PLUS a BY statement. The BY statement forces SAS to process observations paying attention to values of the variables mentioned in the BY statement. To use a BY statement, the datasets must be in the order implied by the statement. The observations in both DATA1 and DATA3 are in ascending order of the values of J. Now when SAS selects observations it takes the observation with the lowest value of the BY variable from ANY of the datasets. If more than one dataset has the same value, SAS takes the observation from the first mentioned dataset with that value first. This INTERLEAVES the observations from the two datasets and produces a dataset that is in order of the BY variables. It also creates some special variables (FIRST, and LAST.) that are very useful in processing, but are not covered here.
The MERGE Statement

The merge operation is the combination of observations together to create an observation that contains the variables from two or more datasets. A MERGE statement without a corresponding BY statement, puts all of the first observations in each dataset together to form an observation, all second, third.... While there are uses, when there is no concern about the matching, the more normal merge uses a BY statement to match observations from each dataset.

If we had demographic dataset for a class of students and another with test scores; without a BY statement we could have chaos. If the first person in the demographic file was not in the test file, every merge would be off. Instead, using a BY statement, SAS takes care of matching the observations together. The datasets MUST be in order of the BY variables. (EXAMPLE 4). The merge without a BY statement has put the first observation in DATA2 (J=1) with the first in DATAl (J=1), the second in DATA2(J=3) with the second in DATAl(J=2), etc.

Note the values of J and WHERE. These variables exist in more than one dataset. When a variable appears in more than one dataset, it value in the output dataset, comes from the last mentioned dataset that contains that variable. SAS moves the observation from the first dataset named into the data vector, then moves the observation from the next dataset in. Thus when variables occur in more than one, the last move of a variable determines its value. If the first dataset has a non missing value of a repeated variable it is moved in, if the last dataset containing the repeated variable has a missing value, SAS doesn't care, it moves in that value (missing). THE VALUE OF EVERY VARIABLE IS IT'S VALUE FROM THE LAST DATASET THAT IT OCCURS IN.

The next example is the same merge with a BY statement. The BY variable is J. Now SAS looks at all of the observations and moves the observation with the lowest value of the BY variable(s) into the data vector, going from left to right. For J=1, it moves an observation from DATA1 then merges the matched observation from DATA2. For J=2, it moves the second observation from DATA1, but there is no match in DATA2 SAS completes processing the observation and continues for J=3.

If a dataset has more than one observation with the same BY values, SAS will generate one output observation for each of the repeats. Note J=3 in the example.

If one dataset does not have the same number of repeats SAS, retains the values for the variables from that dataset in the data vector. When the BY value changes all variables are set to missing and the process starts over.

The next example merges DATAl, DATA2 and DATA3. Note the value of WHERE in observations from DATAl with no matches in either dataset, versus observations with a match with DATA2 only, versus a match with DATA2 and DATA3.

The UPDATE Statement

The UPDATE statement is used to update a master dataset from a dataset containing changes. You MUST use a BY statement. On the update statement there are only two datasets; first, the master dataset to be updated and second the updates to be applied. The master dataset can contain ONLY one observation for a given BY variable (or set of BY variables), the update dataset can have many repeats for a BY value.

Consider a payroll database with one observation for each individual. A person may come in today and tell you that they have moved and to change their address. They come in tomorrow and tell you that their phone number also changed. They come in the next day and tell you that they moved because they got married. There are three changes to be made to the master database; address, phone number and marital status. If you prepared a transaction on the first day, but have not applied the change yet, you may not be able to recall the transaction and add the next change. And you don't have to. Prepare a transaction for each days updates for that individual.

DATA UPDATES;
INPUT (ID PHONE ADDRESS MARITAL...)
(formats);
CARDS;
1234 774-9876 .
1234 . 12 Main St.
1234 .
PROC SORT; BY ID;
DATA NEW_MAST;
UPDATE OLD_MAST UPDATES;
BY ID;

Now when the updates are applied, SAS handles the updates just the way you want it. It moves the master record into the data vector and then one by one moves the NON-MISSING data elements from the update dataset to the master. It does NOT move missing values into the data vector. When all updates have been made, SAS outputs ONE observation to the new master dataset.
The last example UPDATEs DATAl by DATA3. Note that "new" variables may be added through the update process. Missing values for observations do not replace non-missing values in the original.

Summary

SET, MERGE and UPDATE can be used to rearrange data in almost any manner that you wish.

SET:
1) Combine or interleave up to 50 datasets
2) May use a BY statement

Merge:
1) Combines observations
2) Merge 2 to 50 datasets
3) Generally use a BY statement

UPDATE:
1) Updates an observation
2) Updates A master file with an update file
3) Observations in master must be unique.
4) Many updates can be made
5) MUST have a BY statement

****************************
* Three datasets, sufficiently odd, *
* generated to demonstrate *
**************************************************************
DATA; WHERE='FIRST DATA SET';
DO J=1 TO 10; X=J; output; end;
PROC PRINT; TITLE WHERE J X; TITLE3 The SET statement;
TITLE4 SET DATAl;
TITLE5 Selected observations;
END;
PROC PRINT; title "DATAl";

****************************
* examples of the SET statement *
**************************************************************
DATA; SET DATAl;
IF J>5 & J<9;
PROC PRINT;
TITLE WHERE J X;
TITLE3 The SET statement;
TITLE4 SET DATAl;
TITLE5 Selected observations;
**************************************************************
* The next use is the combination of *
* the observations of one dataset with *
* a second (or more) dataset(s). *
**************************************************************
DATA;
SET DATAl DATA3;
PROC PRINT;
TITLE WHERE J X Y Z;
TITLE3 The SET statement;
TITLE4 SET DATAl DATA3;
TITLE5 The combination of 1 and 3;
TITLE5 Note: value of Y and Z;
TITLE6 in observations from DATAl;
**************************************************************
* combining datasets *
* SET with a BY statement *
* interleaving observations *
DATA;
SET DATAl DATA3;
BY J;
PROC PRINT;
TITLE5 With a BY statement;
TITLE6 The SET statement;
TITLE7 SET DATAl DATA3;
TITLE8 The combination of 1 and 3;
TITLE9 Note: value of Y and Z;
TITLE10 in observations from DATAl;
**************************************************************
DATA;
MERGE DATAl DATA2;
PROC PRINT;
TITLE WHERE J X Y;
TITLE3 The MERGE statement;
TITLE4 MERGE DATAl DATA2;
TITLE5 Without a BY statement;
TITLE6 Note: Values of WHERE in first 50 obs.;
**************************************************************
DATA;
MERGE DATAl DATA2;
BY J;
PROC PRINT;
TITLE WHERE J X Y;
TITLE3 The MERGE statement;
TITLE4 MERGE DATAl DATA2;
TITLE5 With a BY statement;
TITLE6 Note: Values of WHERE in merged observations.
**************************************************************
DATA;
MERGE DATAl DATA2 DATA3;
BY J;
PROC PRINT;
**************************************************************
DATA;
UPDATE DATAl DATA3;
BY J;
PROC PRINT;
TITLE WHERE J X Y Z;
TITLE3 The UPDATE statement;
TITLE3 UPDATE DATAl DATA3;
TITLE5 Note: Missing Values;
TITLE6 do NOT replace values;
TITLE7 in the MASTER;