

Point, Set, Match (Merge) – A Beginner’s Lesson

Jennifer Hoff Lindquist, Institute for Clinical and Epidemiologic Research,
Veterans Affairs Medical Center, Durham, NC

The phrase “Point, Set and Match” is used in tennis when the final game winning point is scored. Those terms are also special SAS techniques that can make you a champion SAS programmer. Projects can collect data from a number of different sources. Combining the information into a cohesive data structure is essential for resource utilization. One of the most valuable resources is your time. By combining and collapsing data into a small number of datasets, information can be accessed and retrieved more quickly and already properly linked. Two techniques used with manipulating existing data sets are SET and MERGE.

SET

The SET statement is often used in two ways – copying and appending.

Set – Copy

To avoid corrupting a permanent SAS data set, copy of the data set is desirable. Suppose the permanent SAS dataset In.Source A consists of 5 observation with 4 variables. The syntax to create a copy of the data set is Set <dataset name>.

```
SAS Code:      Data White;
                Set In.SourceA;
                Run;
```

The contents of the White data set are an exact replicate of the original In.Source A data set.

White data set

ID	GRP	AGE	ELIG
1	A	30	Y
2	A	40	N
3	A	50	N
4	A	60	Y
5	A	70	

Set – Append

The SET statement can be used to append or stack data sets.

Let data set Yellow consist of 3 observations with 3 variables

Yellow Data Set

ID	GRP	ELIG
3	B	Yes
5	B	No
6	B	Yes

```
SAS code: Data TwoSets;
            Set White Yellow;
            Run;
```

Any number of data sets could be listed. The white data set contributes 5 observations and the yellow data set tacks on 3 observations. All the variables in the two data sets are included. If any variables are only in one of the data sets, the variable is included in the concatenated dataset. The observations which originated from a dataset without a particular variable has the variable added to the observation with a missing value.

SAS Output:

ID	GRP	AGE	ELIG
1	A	30	Y
2	A	40	N
3	A	50	N

4	A	60	Y
5	A	70	
3	B	.	Yes
5	B	.	No
6	B	.	Yes

The observations are simply stacked starting with the data set listed first. SAS then continues tacking on the observations to the “bottom” of the list for each data set listed in the SET statement. This is especially useful when consolidating data sets with mutually exclusive records but the same variables.

MERGE

MERGE-GENERAL

Joining records “side by side” instead of stacking is another data consolidation technique. Many times your want to create a data set with one observation per patient/person. A merge statement then is more applicable. The remainder of the paper will be devoted to discussing the various types of merges – One to One Merge, Match Merge, One to Many Merge and Many to Many Merge.

MERGE-ONE TO ONE MERGE

The first type of merge is a one to one merge. The accidental or careless use of this merge can produce disastrous results. In all SAS merges a data set listed first (on the left) is overwritten by the corresponding data in the data set on the right. In a one to one merge the observations are conjoined by their relative positions-the first observation with the first observation, etc.

White Data Set					Yellow Data Set		
ID	GRP	AGE	ELIG		ID	GRP	ELIG
1	A	30	Y	↔	3	B	Yes
2	A	40	N	↔	5	B	No
3	A	50	N	↔	6	B	Yes
4	A	60	Y				
5	A	70					

```
SAS Code      Data MergeSet;
                Merge White Yellow;
                Run;
```

In this one to one merge, the values in the first three observations in the white data set are wiped out by the overlapping variables in the yellow data set even though they are NOT referring to the same individuals.

SAS Output

ID	GRP	AGE	ELIG
3	B	30	Yes
5	B	40	No
6	B	50	Yes
4	A	60	Y
5	A	70	

The resulting data set has “lost” patients with ids 1 and 2. The age for patient #3 appears to be 30 when it is actually the age for patient #1. Other errors include the ages for patients 5 and 6. Patient #5 has ages 30 years apart! Due to this potential to lose/corrupt data, the one to one merge is best avoided.

MERGE - MATCH MERGE

A refinement of the one to one merge is the match merge. Using the BY statement, variables are listed which SAS uses to match observations. Data sets must be sorted on the same variables as listed in the match merge BY statement. More than two data sets may be included in the merge statement. More than one variable can be listed on the BY statement. But only one BY statement is allowed for each Merge statement.

```

SAS Code:      Proc Sort data=white;
                  By Id;
Proc sort data=yellow;
                  By Id;
Run;

Data Mergeby;
  Merge white Yellow;
  By ID;
Run;

```

SAS Output

ID	GRP	AGE	ELIG
1	A	30	Y
2	A	40	N
3	B	50	Yes
4	A	60	Y
5	B	70	No
6	B		Yes

If two data sets have variables with the same name, the value of the variable in the data set named last will overwrite the value of the variable in the data set name previously. The ELIG for Patient #3 in the white data set was "N" but in the yellow data set ELIG was "Yes". Since the order in the Merge statement was white then yellow the value in the yellow data set appears in the merged dataset. However, due to the overwrite property this conflict of eligibility status is lost.

The match merge and the one to one merge differ in syntax only in the use of the BY statement. It is (too) easy to inadvertently leave off the BY statement. Results are NOT the same! SAS has acknowledged that this can be a problem. In version 8, there is a system option called MERGENOBY. It has 3 settings – None, Warning, Error. The Warning setting will write a Warning message in the log whenever a merge is performed without a BY statement but will continue processing. The Error setting will write an Error message in the log and halt processing. The None setting – no message is written in the log. I strongly recommend using at least the Warning option.

MERGE -IN Option

A useful option with both the SET and MERGE statements is the IN statement. The syntax is data set name (IN= temporary variable). The temporary variable is assigned a one if the observation came from that data set. The temporary variable is receives a value of zero if the observation is not in that data set. The temporary variable exists only for the length of time it takes to process the observation. It is not accessible after the completion of the data step. If the information will be needed later, a regular variable can copy the value of the temporary variable;

```

SAS Code:      Data MergeSource;
                  Merge White (IN=InWt) Yellow(IN=InYel);
                  By Id;
                  If InWt=1 and InYel=1;
                  *Alternate if InWt=InYel;
                  WtFileInd=InWt;
Run;

```

An intermediate internal processing snapshot shows the values of the temporary variables.

ID	GRP	AGE	ELIG	InWt	InYel	ID	GRP	ELIG
1	A	30	Y	1	0			
2	A	40	N	1	0			
3	A	50	N	1	1	3	B	Yes
4	A	60	Y	1	0			
5	A	70		1	1	5	B	No
				0	1	6	B	Yes

Due to the subsetting if statement the observation must be in both the white and the yellow data sets to make the eligibility requirements for the MergeSource data set. The temporary variables InWt and InYel are not in the resulting data set. The problem remains with the second data set overwriting the first dataset.

Resulting Data set

ID	GRP	AGE	ELIG	WtFileInd
3	B	50	Yes	1
5	B	70	No	1

MERGE - RENAME Option

An option to avoid some of the overwrite problems is to rename the variables in the merge. The syntax is after the dataset name (rename=(old variable=new variable))

```
SAS Code      Data MergeRen;
                Merge White(rename=(ELIG=WELIG)) Yellow(rename=(ELIG=YELIG));
                By ID;
                Run;
```

The original data set supplies the value to the new variable as long as it was the dataset contributing the observation.

SAS Output

ID	GRP	AGE	WELIG	YELIG
1	A	30	Y	
2	A	40	N	
3	B	50	N	Yes
4	A	60	Y	
5	B	70		No
6	B			Yes

By using the rename option, it is possible to detect the inconsistency in patient #3 data.

MERGE - ONE TO MANY MERGE or MANY TO ONE MERGE

A third major category of merges is the One to Many and the closely related Many to One merges. The syntax is the same as the matched merge. However, it is important to know which data set has the “many” observations and which data set has the “one” observation. The order the data sets are listed in the MERGE statement makes a difference. The logistics of the merge is basically the same. The items in the right data set overwrites the data in the left data set.

```
SAS Code:      Data One2Many;
                Merge white green;
                By id;
                Run;
```

Visualizing the data sets side by side will help show what happens.

White Data Set

ID	GRP	AGE	ELIG
1	A	30	F
2	A	40	M
3	A	50	M
4	A	60	F
5	A	70	

Green Data Set

ID	GRP	TYPE
3	C	a
3	C	b
3	C	c
5	C	b
5	C	c

Results of a One to Many Merge

ID	GRP	AGE	ELIG	TYPE
1	A	30	Y	
2	A	40	N	
3	C	50	N	a
3	C	50		b
3	C	50		c

4	A	60	Y	
5	C	70		b
5	C	70		c

The values in variables AGE and ELIG are retained until the value of the BY GRP variable changes.

If the order is reverse, a Many to One merge results in a different data set.

```
SAS Code:      Data Many2One;
                Merge green white;
                By id;
                Run;
```

Looking at the data sets side by side, recall the data on the right overwrites the data on the left.

Green Data Set			White Data Set			
ID	GRP	TYPE	ID	GRP	AGE	ELIG
			1	A	30	Y
			2	A	40	N
3	C	a	3	A	50	N
3	C	b				
3	C	c				
			4	A	60	Y
5	C	b	5	A	70	
5	C	c				

The results of the Many to One Merge

ID	GRP	TYPE	AGE	ELIG
1	A		30	Y
2	A		40	N
3	A	a	50	N
3	C	b		
3	C	c		
4	A		60	Y
5	A	b	70	
5	C	c		

A Many to One merge is possible. However, the values of AGE and ELIG were not retained. The Many to One and the One to Many data sets are different. The differences are highlighted.

Results of a One to Many Merge

ID	GRP	AGE	ELIG	TYPE
1	A	30	Y	
2	A	40	N	
3	C	50	N	a
3	C	50		b
3	C	50		c
4	A	60	Y	
5	C	70		b
5	C	70		c

Be aware the simple change in the order the data sets are listed DOES make a difference.

It is import to know your data. Look at the proc contents before performing merges. Print several observations of the original data sets and the merged dataset. Check and make sure you are getting the results you expect.

MERGE - MANY to MANY MERGE - General

The last category is a Many to Many merge. This type of merge prompts regularly recurring questions on SAS-L, a mail serve list for SAS questions. The problem is the same basic syntax does not yield the desired results for a Many to Many situation!

Green Data Set			Red Data Set	
ID	GRP	Type	ID	CAT
3	C	a	3	10
3	C	b	3	20
5	C	c	5	50
5	C	b	5	60
5	C	d		

The usual DESIRED results are a Cartesian cross product with 10 observations.

ID	GRP	TYPE	CAT
3	C	a	10
3	C	b	10
3	C	c	10
3	C	a	20
3	C	b	20
3	C	c	20
5	C	b	50
5	C	c	50
5	C	b	60
5	C	c	60

However, the expected SAS code does NOT produce the above results.

```
SAS Code:      Data Many2ManyERROR;
                Merge red green;
                By Id;
                Run;
```

Results

ID	GRP	TYPE	CAT
3	C	a	10
3	C	b	20
3	C	c	20
5	C	b	50
5	C	c	60

Possible Solutions include using a SQL procedure or manipulate the dataset with the POINT command.

MERGE – MANY to MANY USING SQL

The SQL procedure implements the Structured Query Language. Using proc SQL a Cartesian cross product data set can be produced.

```
SAS Code:      Proc SQL;
                Create table manySQL as
                  Select *
                    From green, red
                   Where green.id=red.id
                  ;
                Quit;
```

Explanation of SQL code

The phrase "Create table many SQL as" creates a data set named manySQL, storing the results of the query expression.

The code "Select *" includes all the variables all of the data sets listed in the next snippet of code. An alternative is to name the variables you want to keep in the data set.

The names of the source data sets are identified with "From green,red".

The instructions "Where green.id=red.id" states the condition.

POINT in a MANY to MANY MERGE

Instead of using Proc SQL, it is possible to create the data set in a data step. This is accomplished through accessing observations in one data set sequentially and the observations in the other directly using the POINT= statement.

The vast majority of my work I process data sequentially. That is accessing the observations in the order in which they appear in the physical file. Usually every observation needs to be examined, and processed so sequential access is adequate. When working with small datasets sequential access is not a problem.

On occasion it is advantageous to access observations directly. You can go straight to a particular observation without having to handle all of the observations that come before it in the physical file. The POINT= option with the SET statement will communicate to jump to a particular observation. Suppose you had a large data set with 1 million observations named BigSet and you knew ELIG was missing for some observations in the last 100 observations. Instead of sorting or processing the 999,900 other observations you can go directly to the last 100 observations to identify those with missing ELIG values.

```
SAS Code:      Data FindMissing;
                Do I = 999900 to 1000000;
                  Set BigSet point=I;
                  If ELIG="" then output;
                End;
              Stop;
Run;
```

When you use the POINT= option, a STOP statement must be included. If the STOP is inadvertently left off, a continuous or endless loop occurs. You are pointing to specific observations and SAS never will find/read the end of file indicator.

The Many to Many Merge data step solution using the Point option is given below. The Green data set is being processed sequentially. For each observation in the Green data set each observation in the Red data set is accessed directly in the Do Loop. The values read with the SET statement are automatically retained until another observation read from that data set. So each observation of the Red dataset is paired with the retained observation from the Green data set. If the tempId variable in the Green data set matches the Id variable in the Red data set then the observation is outputted to the Cross product data set.

```
Data CrossProduct (drop=tempId);
  Set green(rename=(id=tempId));
  NumInRedSet=4;
  Do i=1 to NumInRedSet;
    Set Red Point=i;
    If tempId=id then output;
  End;
Run;
```

Green Data Set			Red Dataset	
ID	GRP	Type	ID	CAT
3	C	a	3	10
3	C	b	3	20
5	C	c	5	50
5	C	b	5	60
5	C	d		

TempId	GRP	Type	i	ID	CAT	
3	C	a	1	3	10	Output

3	C	a	2	3	20	Output
3	C	a	3	5	50	
3	C	a	4	5	60	
3	C	b	1	3	10	Output
3	C	b	2	3	20	Output
3	C	b	3	5	50	
3	C	b	4	5	60	
3	C	c	1	3	10	Output
3	C	c	2	3	20	Output
3	C	c	3	5	50	
3	C	c	4	5	60	
5	C	b	1	3	10	
5	C	b	2	3	20	
5	C	b	3	5	50	Output
5	C	b	4	5	60	Output
5	C	c	1	3	10	
5	C	c	2	3	20	
5	C	c	3	5	50	Output
5	C	c	4	5	60	Output

Results

ID	GRP	TYPE	CAT
3	C	a	10
3	C	a	20
3	C	b	10
3	C	b	20
3	C	c	10
3	C	c	20
5	C	b	50
5	C	b	60
5	C	c	50
5	C	c	60

CONCLUSION

Trying to locate a particular data element in a large number of small, scattered data sets can be frustrating. Combining data sets with SET and MERGE statements can create data sets which are more comprehensive, cohesive and easier to utilize. The SET statement can be used to copy or append data sets. The MERGE statement used properly pulls together the common data elements for a unit of measurement. Usually a Matched Merge is preferred over a One to One Merge. Using the IN and RENAME options will refine newly created data sets. Many to Many merges are not necessarily intuitive. However, use the proc SQL or the Data Step point examples as templates will get you started on the right path. Combining and manipulating data sets does take a certain amount of skill. But just like tennis, with practice, POINT, SET, and MATCH (Merge), will become part of your winning game set.