Missing Secrets

Robin Jones, Westat
Ian Whitlock, Westat

ABSTRACT

This paper will attempt to explain the secrets to setting SAS variables to missing. Categorizing SAS variables into four types provides a systematic way to unlock such secrets. The four types of variables are 1) unnamed variables, 2) automatic variables, 3) variables from SAS data sets, and 4) all other variables. The rules governing these four types of variables will be discussed. Two major points will also be emphasized; 1) SAS distinguishes between setting a value to missing at the top of the data step loop from setting a value to missing when data is not read; 2) SAS reads in a record only once. Grouping rules according to variable types provides a clearer understanding of when SAS sets variables to missing.

RETAIN

The SAS command, RETAIN, does not mean that a variable must retain its value, i.e. never change. It is simply a request to not set the variable to missing automatically at the top of the implied loop of the DATA step. It does not even mean that the variable may not be set to missing at special times by the system. It is the purpose of this paper to state the rules governing how the system will set or change variable values. The programmer can always make assignment statements or execute other commands that will change the value of a variable. We will not cover such changes. It is the automatic system activity that we want to explicate.

In the following section we look at one of the consequences of the fact that variables coming from SAS data sets are automatically retained.

THE MERGE MYSTERY

Let's say you are merging two data sets, TEST_NUMBERS and FIRST_SCORES, where the data are given by the following steps.

```
Data test_numbers ;
  Input id $ num ;
Cards ;
Jan 1
Jan 2
Jan 3
Tom 1
;

data  First_Scores ;
  input id $ score ;
cards ;
Tom 60
;
```

We have to merge on ID to pick up the first test score for each ID in TEST_NUMBERS. However, Jan is not listed in FIRST_SCORES. Thus if we simply merge by ID then Jan's three test scores will remain missing, as expected.

Now, by other means we learn that Jan's second score is 95, so we add code to specially assign 95 to Jan's second score.

```
data tests ;
  merge test_numbers first_scores ;
  by id ;

  if id = "Jan" and num = 2 then
    score = 95 ;
run ;
```

The code does not work as revealed by the following print.

Obs	ID	Num	Score
1	Jan	1	
2	Jan	2	95
3	Jan	3	95
4	Tom	1	60

Look at Jan's test scores. The first is missing as it should be, and the second is 95 as we assigned it. But why is her third score also 95 instead of missing? What happened?

Because of the BY statement, Jan is still part of the by group processing, and the variable SCORE was part of that processing. At most one record is read from First_SCORES for each by group since there is at most one first score per ID. Remember records are read ONLY ONCE and the data are then retained. At the top of the by group Jan's score was set to missing because no data is contributed from FIRST_SCORES. Variables from SAS data sets are retained so when Jan's second score was changed to 95 her third score also retained this 95 value. It is important to understand that SAS does not read from FIRST_SCORES three times. SAS reads records only once.

In the section "VARIABLES FROM SAS DATA SETS", we will closely examine the rules regarding when and how data values are set to missing.

VARIABLE CLASSIFICATION

In order to give rules for when variable values are set to missing we classify DATA step variables into four types:

1. Unnamed variables.
2. Automatic variables
3. Variables from SAS data sets
4. All other variables

UNNAMED VARIABLES

Unnamed variables come from arrays defined with the key word _TEMPORARY_. The variables are unnamed since they can only be referred to by their array reference. We will call arrays defined in this manner temporary arrays. The system rules governing changes to unnamed variables are:

1. Unnamed variables are initialized once at the beginning of execution to missing; unless explicitly assigned a value.
2. These variables are always retained, i.e. once initialized the system will never reset them to missing; so all changes are the responsibility of the programmer.

Remember that retain does not mean the value never changes, it simply means that the variable will not be set to missing automatically at the top of each loop of the DATA step.

Temporary arrays are unnamed variables. These variables are not part of the logical PDV and their storage units are not written to the output data file.

Here is an example of how to make a temporary array without explicitly assigning values use:

```
Array a (10000) _temporary_ ;
```

Here is an example with an explicitly assigned value:

```
Array a (10000) temporary  (0);
```

These 10000 temporary array elements have an initial value of 0.

AUTOMATIC VARIABLES

1. Automatic variables are initialized once at the beginning of execution to a value that is not missing.
2. The values are retained.
3. The values are reassigned by the system at an appropriate time determined by the request for the automatic variable.
4. It is possible to change the value of an automatic variable using an assignment statement (or some explicit action to change the value), but it is usually not a good idea to do this.

There are two automatic variables present in all data steps _N_ and _ERROR_. _N_ is the counter for the number of implied iterations in a DATA step. _ERROR_ is the indicator for execution time errors in the DATA step.

Automatic variables like unnamed variables are not written to the output file.

Here are the rules for these variables.

For _N_:

Initialized to 1

Copied at the top of each implied loop of the DATA step from a secure system counter.

For _ERROR_:

Set to 0 at the top of each implied loop of the DATA step.

Set to 1 when a system error occurs.

Can be set to 1 by the programmer to indicate an execution time error.

When a BY statement is present in a DATA step, two automatic variables are created for each variable, VAR, in the BY statement. They are FIRST.VAR and Last.VAR. Typically they are called first dot and last dot variables. If a BY statement is present they are initialized to 1 and reset when the corresponding SAS input statement is executed. (They are initialized to 0 when created without a BY statement.)

Automatic variables can also be created by options on various SAS statements. Here is an example of IN= automatic variables from the MERGE statement.

```
Data _null_ ;
  Merge a (in=in_a)
        b (in=in_b) ;
  By id ;
Run ;
```

There will be 2 additional variables in the PDV called IN_A and IN_B. Both of these variables are initialized to 0 and reset to 0 when the MERGE statement executes at the beginning of a BY group. They are reset to 1 each time a record is read from the corresponding data set. Like all automatic variables their values are retained.

VARIABLES FROM SAS DATA SETS

Below are 5 rules governing how the system changes values for variables coming from SAS data sets: The rules are necessary because defining the RETAIN

statement alone can not sufficiently explain how values are assigned.

1. Variables from a SAS data set are initialized once at the beginning of execution to missing.
2. If the variable comes from a SAS data set participating in BY processing, the variable will be set to missing at the beginning of each by group. This occurs during the execution of the SET, MERGE, UPDATE, or MODIFY statement.
3. If the variable comes from a SAS data set participating in a SET statement, the value will be set to missing every time the buffer is switched for that SET statement. This action takes place during the execution of the SET statement.
4. If the variable comes from a SAS data set it's value will change according to the value read each time a record is read from corresponding data set.
5. If a user makes an assignment (or some explicit action to change the value), the value will change.

Let's look at an example that demonstrates how the rules apply using the data sets TEST_NUMBERS and FIRST_SCORES created in the section "THE MERGE MYSTERY".

```
data adhoc ;
  var = 66 ;
run ;

data scores ;
  if _N_ = 1 then set adhoc ;
  set test_numbers first_scores ;
  by id ;
  if id = "Jan" and num = 2 then
     score = 95 ;
run ;

proc print data = scores ;
run ;
```

Obs	Var	ID	Num	Score
1	66	Jan	1	
2	66	Jan	2	95
3	66	Jan	3	95
4	66	Tom	1	
5	66	Tom		60

The VAR variable was read from data set ADHOC's buffer. The buffer was switched and the VAR value was set to missing at the top of the first iteration of the implied loop. The value 66 was read from ADHOC only once and was not explicitly changed. The value did not change because it is not part of the by group processing. Therefore, the value VAR was never set to missing at the beginning of the by group.

Tom's value for the NUM variable is missing on observation #5 because there is indeed a record in First_Scores. At _n_ = 5 the buffer is switched to First_Scores, hence, following rule #3, Tom's NUM value is set to missing.

You can see that Jan's first score is a missing value. In Jan's by group, the initial value for the variable SCORE is set to missing. SAS attempts to read the data. There are no records for Jan in First_Scores. SAS attempted to read in a value ONCE and will not try to read from First_Scores again for Jan.

Were the IF statement not in the code, all of Jan's three scores would be missing. The missing data would have been retained after the system set SCORE to missing at the beginning of the BY group. The assignment in the IF statement set Jan's score to 95 on her second record and the score of 95 was retained for all of her subsequent records.

ALL OTHER VARIABLES

The remaining variables follow these rules.

1. They are set to missing at the top of each implied DATA step loop.
2. The values are retained if and only if they are mentioned in an explicit RETAIN statement.

These variables typically are created by assignment statements or function calls. For example:

```
data old ;
  x = 8 ;
  y = 4 ;
  output ;
  x = 4 ;
  y = 2 ;
  output ;
run ;

Data new ;
  put _ALL_ ;
  Set old ;
  Var = x / y ;
Run ;
```

Log output:

```
x=. y=. Var=. _ERROR_=0 _N_=1
x=8 y=4 Var=. _ERROR_=0 _N_=2
x=4 y=2 Var=. _ERROR_=0 _N_=3
```

In the above example, the variable VAR is an assigned variable. The variables X and Y are retained. The variable VAR is not retained across iterations, hence, the values are set to missing at each iteration of the implied DATA step loop.

CONCLUSION

The rules for knowing how the SAS system changes variables are not hard, but few can precisely state all the rules. Since the appropriate rule depends on the context in which the variable is created the SAS documentation for these rules is spread all over the manual. The secret to understanding lies in the classification of the variables into four distinct groups. When the rules are understood you have a better understanding of how the MERGE statement sometimes results in mysterious and unintended values. They also help in understanding other by processing results.

REFERENCES

SAS Institute Inc., SAS Language: Reference, Version 6, First Edition, Cary, NC: SAS Institute Inc., 1990.

CONTACT INFORMATION

Ian Whitlock
Westat
1650 Research Boulevard
Rockville, MD 20850

E-mail: IanWhitlock@Westat.com

Robin Jones
Westat
1650 Research Boulevard
Rockville, MD 20850

Phone: (240) 453-2996
Email: RobinJones@Westat.com