Effective Use of RETAIN Statement in SAS® Programming

Yi Zhao
Merck & Co. Inc., Upper Gwynedd, Pennsylvania

INTRODUCTION

The RETAIN statement allows values to be kept across observations enabling complex data manipulation. In SAS data step, it is quite straightforward to do variable comparison or data calculations within observations. However, sometimes it is necessary to perform calculations across observations. The RETAIN statement could be used to keep a data value from the current iteration of the data step to the next data step. Otherwise, SAS automatically sets such values to missing before each iteration. However, the RETAIN statement, if not used wisely, may result in unexpected and often unnoticed data processing errors. This paper presents several practical usages of the RETAIN statement in data processing.

CARRY OVER VALUES FROM ONE OBSERVATION TO ANOTHER

Suppose we have a lab dataset with lab tests performed in scheduled as well as unscheduled visits. The visit number for an unscheduled visit is recorded as 99. For analysis purposes, we need to convert 99 to a true visit number with a decimal indicating its unscheduled nature. This task requires us to carry over the visit number from one observation to the next. RETAIN statement is used to solve this problem effectively; following is the SAS code for the RETAIN statement:

```
proc sort data = lab;
    by subjid test_dt;
run;

data lab_new;
    set lab;
    by subjid;
    retain visit_r 0;
    if first.subjid then visit_r = visit;

    if visit = 99 then visit = visit_r + 0.1;
    visit_r = visit;
run;
```

The dataset is sorted in ascending order by test date per subject so that the earliest lab test record is read into the Program Data Vector (PDV) first. The visit number is then assigned to the retained new variable VISIT_R. The RETAIN statement prevents it from being set to missing from one iteration of the data step to the next. The next 'if --- then' statement checks the visit value and adds 0.1 to the previously retained visit number if the value is 99 and assigns the new value back to variable VISIT. Finally, the retain variable VISIT_R is updated to keep the most recent visit number in order to carry over to the next observation. The output dataset lab_new has a new visit number assigned for each unscheduled visit based on its previous visit number.

COMPARE VALUES ACROSS OBSERVATIONS

The next example is to create a derived dataset called ABNORMAL with a flag variable FLAG to indicate if a lab test result is abnormal; following is the code:

```
proc sort data = lab;
    by subjid test_dt;
run;

data abnormal;
    set lab;
    by subjid;
    retain nadir flag;

    if first.subjid then do;
        nadir = result;
        flag = 0;
    end;

    if 0 < result < nadir then nadir = result;

    if log10(result/nadir) > 1 then flag = 1;

    if last.subjid and flag = 1 then output;
run;
```

The retained variable FLAG and NADIR are re-set to 0 and the value of RESULT respectively when the first observation for each subject is read in. This step is necessary since the BY statement will not cause the RETAIN statement to automatically re-set for each BY variable (or group of BY variables) and must be re-set in the code explicitly. It is at this step inexperienced SAS programmers are likely to make the mistake of assuming the BY variable(s) will re-set the retained variable(s).

Variable NADIR is compared to the value of RESULT in each row successively. If its value is less than that of RESULT, the smaller value is assigned to NADIR. As a result, NADIR keeps the minimum value of RESULT. Finally, the flag is set based on the minimum value of RESULT retained in variable NADIR.

ASSIGN INITIAL VALUES

The RETAIN statement could be used to initialize values for individual variables, a list of variables, or members of an array. If a value appears in a RETAIN statement, variables that appear before it in the list are initially set to that value. If different initial values are assigned to the same variable by naming it more than once in a RETAIN statement, SAS uses the last value.

There are multiple approaches to assigning initial values with the RETAIN statement, and each approach has its unique usefulness according to the programming needs. The following shows some example variables specified in each RETAIN statement:

RETAIN var1; sets initial value of var1 to missing.
RETAIN var1-var3 0; sets initial values of var1, var2, var3 to 0.
RETAIN var1-var3 (0); only var1 is set to 0, var2-3 are set to missing.
RETAIN var1 0 var2 1 var3 1; sets initial value of var1 to 0, var2 and var3 to 1.
RETAIN var1-var3 (0 1); sets initial values of var1 to 0; var2 to 1, var3 to missing.
RETAIN var1-var3 (1 2 3); or
RETAIN var1-var3 (1,2,3); sets var1 to 1, var2 to 2, var3 to 3. Comma is optional.
RETAIN; or
RETAIN _ALL_; [or _CHAR_, _NUMERIC_]; all variables are retained. The difference is that the first one causes all variables to be retained, whereas the second one affects only those variables defined before the RETAIN statement.

If a variable name is specified only in the RETAIN statement and an initial value is not specified, the variable is not written to the dataset, and a note stating the variable is uninitialized is written to the SAS log. If an initial value is specified, the variable will be written to the output dataset.

DETERMINING COLUMN ORDER IN OUTPUT DATASET

Sometimes we want to arrange the variables in the output dataset in a certain order. For example, identification variables go before other variables, and we can achieve this goal by several approaches. The most obvious approach is to use ATTRIBUTE or LENGTH statement. However, the specific values such as length, format or label have to be specified when used. A more simple approach is to use the RETAIN statement; following is an example:

```
data out.lab;
        retain subjid test_name test_dt test_result;
        set lab;
        /* other statements */
Run;
```

We must make sure the RETAIN statement occurs before the SET statement. Consequently, the variables in the RETAIN statement are positioned first (leftmost) in the PDV and the output data set.

CONCLUSION

- We use the RETAIN statement when we want SAS to preserve a variable's value from the previous iteration of the DATA step. As a result, we could carry over the value of variables from one observation to another to perform calculations or compare values across observations.
- There are multiple approaches to using the RETAIN statement to assign initial values for variables based on programming needs.
- The RETAIN statement, if suitably positioned, is a straightforward way to arrange the order of variables in a SAS dataset to meet external requirements.
- In summary, the RETAIN statement can be used in novel ways to circumvent problems that commonly arise in the process of programming.

REFERENCES

Gorrell, Paul "The RETAIN Statement: One Window into the SAS Data Step" *Proceedings of the 1999 Annual NorthEast SAS Users Group Conference*, BT064, 1999.

AUTHOR CONTACT INFORMATION

Yi Zhao
Senior Scientific Programming Analyst
Merck Research Laboratories
UG1CD-38
PO Box 1000
North Wales, PA 19454
Phone: 267-305-7672
Email: yi_zhao@merck.com