Paper 1751-2014

Ordering Columns in a SAS® Dataset: Should you really RETAIN that?

Andrew Clapson, Statistics Canada

ABSTRACT

When viewing and working with SAS® data sets - especially wide ones - it's often instinctive to rearrange the variables (columns) into some intuitive order. The RETAIN statement is one of the most commonly cited methods used for ordering variables. Though RETAIN can perform this task, its use as an ordering clause can cause a host of easily missed problems due to its intended function of retaining values across DATA step iterations. This risk is especially great for the more novice SAS programmer. Instead, two equally effective and less risky ways to order data set variables are recommended, namely, the FORMAT and SQL SELECT statement

INTRODUCTION

Nearly every method of arranging variables in a dataset relies on the fact that SAS orders columns based on the order in which the variables are declared or defined. As such, declarative statements such as RETAIN will serve to order columns in a dataset when they are placed prior to the SET statement. However, some methods are better choices than others: namely the FORMAT and SQL SELECT statements, as they do not have the same potential for error that comes with the use of RETAIN for this purpose.

INTENDED USE OF THE RETAIN STATEMENT

The RETAIN statement functions to (surprise!) retain values from one iteration of the datastep to another. Keeping in mind that the datastep is an implicit 'do' loop, this statement is invaluable in many situations. For instance, if we want to maintain a running count or sum of a variable in a datastep, RETAIN is our best friend. As an example, here we compute total sales by region for a fictional shoe store:

```
proc sort data = SASHELP.Shoes;
       by Region Subsidiary;
run;

data WORK.RegionSales_1;
set SASHELP.Shoes;
       by Region;
              retain TotalSales;
              format TotalSales DOLLAR12.;
              if first.Region then TotalSales = Sales;
                     else TotalSales = sum(TotalSales, Sales);
run;
```

Without the RETAIN statement the value of "TotalSales" would be reset with each datastep iteration and we would not be able to compute a running total. Note also that the RETAIN statement is placed *after* the SET statement – if RETAIN was placed before SET, it would function identically but have the additional effect of declaring the "TotalSales" variable before all other variables and thus establishing it as the first column in the output dataset.

WHAT CAN GO WRONG?

SAS arranges columns in a dataset according to the order in which the variables are declared or defined, so any variables declared prior to the SET statement will be placed first in the resulting output dataset. However, unless the sole purpose of a datastep is to re-order variables, the use of RETAIN for this purpose can lead to unwanted side effects when the value of any 'retained' variable is carried over from one datastep iteration to another. In the following (admittedly naive) example, this behavior leads to a rather serious data error:

```
proc sort data = SASHELP.Class;
	by Sex descending Name Weight;
run;

data WORK.ClassHeights_1 (keep = Name Height Category);
	retain Name Category Height;
set SASHELP.Class;
	if (Height < 59) then Category = 'S';
		else if (Height > 59) AND (Height < 67)  then Category = 'M';
		else if (Height > 67) then Category = 'L';
run;
```

Experienced SAS programmers will see the problem right away: due to the non-exhaustive IF/ELSE conditions, individuals with a height of 59 or 67 inches are not explicitly assigned a value for "Category". The inclusion of the RETAIN statement, however, means that height values of 59 and 67 in fact *will* be assigned to a category - they are assigned the value of "Category" that was retained from the previous datastep iteration. This leads to arbitrary and likely erroneous results, especially if the data is unsorted or sorted in an unexpected way, as above. In this case, they are assigned categories of 'L' and 'S', respectively, where instead we might have expected a missing value.

Of course, the above error would be corrected by the inclusion of a catch-all ELSE statement, but this is only a simple case; with several levels of IF conditions and variable assignments in a single datastep it's entirely possible to have a similar error occur, especially if you rely on missing values for OUTPUT statements or WHERE conditions later in the code. Having personally encountered the above error multiple times, it seems reasonable to warn against it and to suggest safer alternatives for arranging variables.

TWO ALTERNATIVES FOR ORDERING COLUMNS

Though RETAIN arguably *does* accomplish the task of ordering variables in a dataset, it has a rather serious potential downside that can lead to errors that are both serious and silent: the worst kind of error to have! Instead, consider as alternatives either the FORMAT or SQL SELECT statements, two safer ways of ordering columns.

The FORMAT statement, when placed before the SET statement and followed by a variable list (no formats need be specified), will arrange columns in a dataset:

```
data WORK.ClassHeights_2 (keep = Name Height Category);
	format Name Category Height;
set SASHELP.Class;
	if (Height < 59) then Category = 'S';
		else if (Height > 59) AND (Height < 67)  then Category = 'M';
		else if (Height > 67) then Category = 'L';
run;
```

Note: the FORMAT statement must be placed before the SET statement. Placing FORMAT after the SET statement will reset the formats for any variable included in the ordering list.

When using the SQL procedure, the order in which variables are listed in the SELECT statement will establish the variable order for the resulting output dataset[1]:

```
proc SQL;
	CREATE TABLE WORK.ClassHeights_3 AS
		SELECT Category, Height, Name
			FROM WORK.ClassHeights_2;
QUIT;
```

CONCLUSION

Though the RETAIN statement can be used to order columns in a SAS dataset, its intended function of retaining values across datastep iterations can make it a poor choice for this purpose. Instead, the use of either the FORMAT statement for datasteps or the SELECT statement for PROC SQL should be encouraged, as it has the same effect without the corresponding risk of serious data errors.

---

[1] This use of the SELECT statement requires all desired variables to be included in the list. And while the use of the wildcard operator '*' is permitted here, it will result in warnings being printed to the log.

ACKNOWLEDGMENTS

The author would like to thank John Ladds and Art Tabachneck for their valuable input and advice.

CONTACT INFORMATION

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

Name: Andrew Clapson
Organization: Statistics Canada
Address: Jean Talon Building, 170 Tunney's Pasture
City, Province, Postal Code: Ottawa, ON | Canada | K1A 0T6
Work Phone: 1-613-951-4308
Email: andrew.clapson@statcan.gc.ca