Paper 060-31

The REPORT Procedure's Temporary Variable: What is it? Why Do I Care?

Mike Molter, Howard M Proskin & Associates, Rochester, NY

ABSTRACT

PROC REPORT can do it all. That was my reaction after I learned about all the tasks that I could perform with this procedure. Not only can I display and summarize data as I can with other procedures, but I can also customize how the output looks. Even better than that, I can perform certain DATA step-like tasks such as creating data or defining attributes with DATA step-like processing such as conditional or iterative looping logic. However, as I began playing with some of these features, I soon discovered that the underlying file on which the report is based is not what I had expected, making conditional execution of statements based on their values unpredictable. Luckily, my countless journeys through SAS OnlineDoc® had exposed me to temporary variables. Perhaps an under-documented feature, temporary variables give us the opportunity to, in effect, make the file on which the report is based more like the more familiar data set. Conditional processing within COMPUTE blocks based on these variables becomes much more intuitive and predictable.

INTRODUCTION

While the REPORT procedure offers functionality such as displaying, grouping, and summarizing data similar to other PROCs, it is the COMPUTE block that sets this procedure apart from the rest. Unlike anything found in almost any other PROC, the COMPUTE block allows DATA step logic and statements such as IF-THEN-ELSE, ARRAY, iterative DO loops, and assignment statements to perform such tasks as creating new variables (items) in the report, defining display attributes such as color and font size, and creating custom summaries. This is made possible through the creation by the SAS® system of a temporary file based on the needs specified in the PROC. The temptation is to assume that the structure of this file and the processing of its records is the same as that of the DATA step, but without knowledge of this file and how the report is built, unexpected results can occur.

THE TEMPORARY FILE

One of the biggest differences between DATA step processing and COMPUTE block processing is in references to other variables and their values. Just as the DATA step processes one record at a time and temporarily stores variable values in the program data vector (PDV) until the observation is output, PROC REPORT also processes one record from the temporary file at a time in a temporary memory buffer. The DATA step, however, populates all the variables in the PDV before executing the other statements. PROC REPORT, on the other hand, processes variables in the order in which they are named on the COLUMN statement. For that reason, variables referenced in a COMPUTE block cannot refer to variables listed to their right in the COLUMN statement. This begins to show us that in the COMPUTE block of PROC REPORT, DATA step logic has to be implemented based more on how the report is laid out rather than how we might imagine the temporary file is structured or built.

With more of an emphasis on the report layout, we now consider how to reference values of GROUP variables. The use of temporary variables to accomplish this task will be the focus of the remainder of this paper. We know that the observations in a data set that are represented by a particular value of a GROUP variable in the report all have the same value for that variable in the data set. This is not true in the report. We know that in the report (unlike the results of the PRINT procedure), each unique value of a GROUP variable is displayed only once in the column. As other report variables to the right change values while this unique value stays the same in the data set that feeds the PROC, empty space fills the column underneath the unique value in the report. Not only is this true of the displayed output, but it is this structure on which data step logic in the COMPUTE block is based. The Online Documentation does not give us details about what the temporary file looks like, and so the picture we paint of this file should resemble closely the report itself. This has significant implications on how we make reference to GROUP variable values in a COMPUTE block. For example, suppose that a data set contains a variable *x* whose value is 1 for a set of observations, and that for some of these observations, the value of *y* is 1 and for others, the value of *y* is 2. Suppose now that we summarize a numeric variable using PROC REPORT using *x* and *y* as GROUP variables. Then the value of *x* when *y* is 1 is 1, but the value of *x* when *y* is 2 is *missing*. We illustrate the importance of this in the following example.

THE GROCERY EXAMPLE

Consider the following PROC REPORT code.

```
ods rtf file='c:\report1.rtf';
proc report data=grocery nowindows;
column sector department sales ;

 define sector     / group ;
 define department / group ;
 define sales      / analysis sum format=dollar9.2 ;
 where sector eq: 'N';
 title 'Report for Northeast and Northwest Sectors';
 run;
 ods rtf close ;
```

The output that this code generates is in Table 1.

Sector	Department	Sales
Northeast	Canned	$840.00
	Meat/Dairy	$490.00
	Paper	$290.00
	Produce	$211.00
Northwest	Canned	$1,070.00
	Meat/Dairy	$1,055.00
	Paper	$150.00
	Produce	$179.00

Table 1

Now suppose that we want to emphasize the sales amount for the Canned department by making it bold. This can be achieved with CALL DEFINE as follows.

```
compute sales ;
if department eq 'Canned' then
  call define(_col_,'style','style=[font_weight=bold]');
endcomp;
```

CALL DEFINE does not always have to be embedded inside a COMPUTE block, but because it is being executed conditionally, in this case it does. Of course in this case, two sales amounts, $840.00 and $1,070.00 will displayed in bold. Suppose we only want to emphasize the amount for the Canned department in the Northeast region. Now the code is as follows.

```
 compute sales ;
if sector eq 'Northeast' and department eq 'Canned' then
call define(_col_,'style','style=[font_weight=bold]');
endcomp;
```

This produces the table in Table 2.

Sector	Department	Sales
Northeast	Canned	**$840.00**
	Meat/Dairy	$490.00
	Paper	$290.00
	Produce	$211.00
Northwest	Canned	$1,070.00
	Meat/Dairy	$1,055.00
	Paper	$150.00
	Produce	$179.00

Table 2

Now suppose we want to emphasize the sales amount for the Meat/Dairy department of the Northeast sector. Intuition might tell us to run the following COMPUTE block.

```
compute sales ;
  if sector eq 'Northeast' and department eq 'Meat/Dairy' then
  call define(_col_,'style','style=[font_weight=bold]');
  endcomp;
```

All we have done is changed 'Canned' to 'Meat/Dairy'. This seems natural because we know that in the data set that feeds the PROC (GROCERY), observations do exist where 'Northeast' is the value of SECTOR and 'Meat/Dairy' is the value of DEPARTMENT. The output suggests something different. By running this code, the result is the same as that of Figure 1 above in which nothing is bold. This is because the value of SECTOR is missing when the value of DEPARTMENT is 'Meat/Dairy'. This suggests to us that the structure of the temporary file is more like that of the report itself than the data set that fed it. Of course changing the criterion from "sector eq 'Northeast' " to "sector eq '' " does not help because the Meat/Dairy department in the Northwest sector also has a missing value for SECTOR for this department, and so both sales amount would be displayed in bold.

THE TEMPORARY VARIABLE

The solution to this problem is the use of a temporary variable, a feature specific to PROC REPORT. The concept of a temporary variable is not new to us. Variables named with the END= option on the SET statement in the DATA step or with the DATA step option IN= are also temporary. These are variables a programmer has the option to create that are not part of any data being read and will not be part of the resulting data set. Temporary variables in PROC REPORT are created by the programmer in a COMPUTE block, and will not be part of the final report. They also do not appear in the COLUMN statement nor in any DEFINE statement. The key advantage they provide is that once the value is set, it is retained until explicitly changed.

For this example, we create the temporary variable at a location in the report where we know that the value of SECTOR is not missing. One option is when we know the value of SECTOR changes. Below is the code with which we started.

```
ods rtf file='c:\report1.rtf';
proc report data=grocery nowindows;
column sector department sales ;

define sector     / group ;
define department / group ;
define sales      / analysis sum format=dollar9.2 ;
where sector eq: 'N';
title 'Report for Northeast and Northwest Sectors';
```

We now add the following COMPUTE block.

```
compute before sector ;
temp=sector ;
endcomp;
```

The variable *temp* is a temporary variable. Notice that it does not appear in the COLUMN statement, nor in a DEFINE statement. Additionally, it is known to all COMPUTE blocks. We now emphasize the sales amount for the Meat/Dairy department in the Northeast, conditional on *temp*.

```
compute sales ;
if temp eq 'Northeast' and department eq 'Meat/Dairy' then
call define(_col_,'style','style=[font_weight=bold]');
endcomp;
```

OTHER USES

References to GROUP variable values are not the only useful purposes of temporary variables. We have all appreciated the availability of the RETAIN statement in the DATA step. When building a report, PROC REPORT first calculates summary statistics for all analysis variables at each possible level of summarization. The ability to access these and hang on to them allows us to create percentages with denominators of our choice. The TABULATE procedure provides us with notation to specify denominators. With PROC REPORT, we create a temporary variable immediately before the value of the variable whose sum will be the denominator changes, and then create the percentage variable in a separate COMPUTE block. The code is below.

```
ods rtf file='c:\report1.rtf';
proc report data=grocery nowindows;
column sector department sales percentage;

define sector     / group ;
define department / group ;
define sales      / analysis sum format=dollar9.2 ;
define percentage / computed format=percent8.2;
where sector eq: 'N';
title 'Report for Northeast and Northwest Sectors';

compute before sector ;
temp=sector ;
denominator=sales.sum;
endcomp;

compute percentage ;
percentage=sales.sum/denominator ;
endcomp;

compute sales ;
if temp eq 'Northeast' and department eq 'Meat/Dairy' then
call define(_col_,'style','style=[font_weight=bold]');
endcomp;
```

In this example, the temporary variable *denominator* is created when the value of *sector* changes, and is used in the calculation of *percentage* in the next COMPUTE block.

CONCLUSION

I have always found that the Online Documentation has been a helpful source of information, particularly for PROC REPORT which I learned primarily through this resource. The section labeled HOW PROC REPORT BUILDS A REPORT has helped me to learn the details of how logic in this PROC is processed. There are, however, pieces of information for which the documentation could be improved. First, this is one of the only places where temporary variables are mentioned, and few examples exist to illustrate them. Second, no details are provided on the structure of the temporary file and how PROC REPORT uses it. Third, two examples are provided in this sub-section, the second of which does illustrate the use of a temporary variable and an IF statement that uses its value. Unfortunately, the documentation details only how the first detailed line of the report is built, and then only tells you that the other detail lines are created similarly, passing up the chance to educate you a little about the structure of the temporary file and the missing values of the GROUP variables. It is my hope that this paper has helped to clear up any confusion you may have had, either from reading that documentation or through your own experiences.

CONTACT INFORMATION
Please feel free to contact me with any comments and questions.

Mike Molter
Howard M. Proskin and Associates
Rochester, NY 14612
(585) 359-2420
mmolter@hmproskin.com