

Under the Covers of PROC REPORT

Keith Cranford, Office of the Attorney General of Texas, Child Support Division

Abstract

How does PROC REPORT really work? This paper attempts to answer this question by examining how PROC REPORT processes your data to produce a report. PROC REPORT works similarly to the DATA step, but with some important differences. Both listing and summary reports are used to illustrate these similarities and differences. It is hoped that this understanding will provide new insights into how to use PROC REPORT to its fullest.

General Sequence

PROC REPORT works like a combination of PROC MEANS and the DATA step. The procedure combines much of the summarization capabilities of a procedure such as MEANS or TABULATE with the line by line processing that occurs in the DATA step. This is what gives PROC REPORT its special flexibility.

PROC REPORT works like the DATA step in that it works sequentially through your data. The general sequence of steps is as follows.

1. Data is summarized by the group, order and across variables. All statistics are calculated for the report, including the detail rows and summary lines. These are stored in a temporary file.
2. All temporary variables are set to missing.
3. Rows are constructed line by line.
 - a. Report variables are set to missing.
 - b. Values of the report variables are filled in from left to right. Computed variables are determined by executing the statements in the compute block, while the values for all other variables come from the temporary file.
 - c. For break lines, values are first determined from BREAK or RBREAK statements. If there is a COMPUTE block associated with the break, these statements are then executed.
4. Completed row is written to the report.

Steps 3 and 4 are repeated until all rows of the report are completed. Report variables are variables that are specified in the COLUMN statement. These can come from the input data set or can be calculated in a COMPUTE block. Temporary variables are variables that only appear in a COMPUTE block.

As may be seen from this sequence, PROC REPORT is similar to the DATA step the way it works one row at a time, outputting a row each time through the process. However, two differences can also be noted. First, unlike the DATA step which reads an entire record or observation from an input file or data set before processing, PROC REPORT works one column at a time. This makes the order of the variables in the COLUMN statement very important. For example, a variable should not be used in a COMPUTE block unless it is to the left of the variable being calculated. Secondly, because summarization of the data has already occurred, these statistics are available anywhere within PROC REPORT. This is like having the DATA step start with an initial read of the data to summarize the data, and then read through the data again to process individual records or observations.

This sequence of steps will be illustrated in a series of examples of the various ways PROC REPORT is used.

Listing Report

The simplest example is a listing, or detail, report, where one observation from the input data set is displayed on each row in the report. There may be additional calculated columns, and summary lines may be included. This can be thought of as using PROC REPORT as an enhanced PROC PRINT.

For the examples in this paper, a very simple data set is used. The data set, named TEST, contains one character variable, two numeric variables, and seven observations. This data set is shown in Table 1.

Table 1. Sample Data Set

Group	x	y
A	1	4
A	2	5
B	2	7
B	3	8
B	4	6
C	1	3
C	4	6

The first example shows a listing report displaying all variables from TEST and creating a new column, z, which is the sum of x and y. Also, a summary line is given for each value of the variable Group. The code is:

```
proc report data=test nowd ;
  column Group x y z ;
  define Group / order ;
  define x / sum ;
  define y / sum ;
  define z / computed ;
  break after Group / summarize skip ol ;
  compute z ;
    z=x.sum + y.sum ;
  endcomp ;
  compute after Group ;
    Group='Total' ;
  endcomp ;
run;
```

Following the sequence listed above, PROC REPORT processes this report as follows:

1. **Create summary file.** This file will look something like this.

Group	x	y
A	1	4
A	2	5
Summary for A	3	9
B	2	7
B	3	8
B	4	6
Summary for B	9	21
C	1	3
C	4	6
Summary for C	5	9
Overall Summary	17	39

2. **Initialize temporary variables.** There are none in this example.
3. **Fill in values for first row.**
 - a. **Initialize first row.**

Group	x	y	z
	.	.	.

- b. **Fill in report variables from left to right.**

from the temporary file:

Group	x	y	z
A	.	.	.

Group	x	y	z
A	1	.	.

Group	x	y	z
A	1	4	.

from the compute block:

Group	x	y	z
A	1	4	5

4. **Write the complete row to the report.**

Steps 3 and 4 are repeated for each row in the temporary file. However, the BREAK statement and COMPUTE AFTER statement dictates what happens for the summary rows. First, the BREAK statement indicates that the summarized values are displayed. This causes the summarized report variables x, y and z (in this case the sum of these) to be displayed.

Group	x	y	z
A	3	9	.

The COMPUTE block is executed, computing the value of z by adding x and y.

Group	x	y	z
A	3	9	12

Then the COMPUTE AFTER block is executed, replacing the 'A' in the Group column with 'Total.'

Group	x	y	z
Total	3	9	12

Continuing in this manner produces the final report.

12:47 Tuesday, June 14, 2005

Group	x	y	z
A	1	4	5
	2	5	7
Total	3	9	12
B	2	7	9
	3	8	11
	4	6	10
Total	9	21	30
C	1	3	4
	4	6	10
Total	5	9	14

Considering the above sequence of steps, a few comments are in order. First, the summarization is done whether summary lines are displayed or not. The temporary file contains the summarized data, but these will not be displayed

unless a BREAK or RBREAK statement is included with the SUMMARIZE option, or a COMPUTE AFTER|BEFORE statement is used that includes the summary variables. This can be seen in the example where there is no RBREAK, so the overall summary information is not displayed. Note, also, that because the data is already summarized before the report is generated, the summary lines could precede the detail rows.

If a report variable appears in a COMPUTE block before it is added, a missing value results. For example, if the COLUMN statement above is changed to

```
column Group x z y;
```

PROC REPORT will fill in the first row as follows.

Group	x	z	y
A	.	.	.

Group	x	z	y
A	1	.	.

from the compute block, $z = \text{sum.x} + \text{sum.y}$

Group	x	z	y
A	1	.	.

Group	x	z	y
A	1	.	4

The result is a missing value of z, because when the value of z is determined the value of sum.y is missing.

Summary Report

Another use of PROC REPORT is to produce summary reports. The same sequence of steps is used to produce these as the listing reports described above. The major difference is what is displayed. With summary reports only the summary rows are displayed.

A modification of the first example will help illustrate these points. By changing the usage of Group to "group" from "order" the report is changed from a listing to a summary report. Also, the BREAK is changed to an RBREAK, and the COMPUTE AFTER Group is changed to COMPUTE AFTER.

```
proc report data=test nowd ;
  column Group x y z ;
  define Group / group ;
  define x / sum ;
  define y / sum ;
  define z / computed ;
  compute z ;
    z=x.sum + y.sum ;
  endcomp ;
  rbreak after / summarize ol ;
  compute after ;
    Group='Total' ;
  endcomp ;
run;
```

Following the sequence of steps again, PROC REPORT processes this report as follows.

1. **Create summary file.** This file is exactly the same as before.

Group	x	y
A	1	4
A	2	5
Summary for A	3	9
B	2	7
B	3	8
B	4	6

Summary for B	9	21
C	1	3
C	4	6
Summary for C	5	9
Overall Summary	17	39

2. **Initialize temporary variables.** Again, there are none in this example.
3. **Fill in values for first row.**
 - a. **Initialize first row.**

Group	x	y	z
	.	.	.

- b. **Fill in report variables from left to right.** This is a summary report, so only the summary lines are used.

from the temporary file:

Group	x	y	z
A	.	.	.

Group	x	y	z
A	3	.	.

Group	x	y	z
A	3	9	.

from the compute block:

Group	x	y	z
A	3	9	12

4. **Write the complete row to the report.**

Steps 3 and 4 are repeated until the last line of the report where the RBREAK and COMPUTE AFTER statements determine the values. Reading from the temporary file creates the following initial summary line.

Group	x	y	z
C	17	39	56

The COMPUTE AFTER block alters the value of Group.

Group	x	y	z
Total	17	39	56

The final report is:

12:47 Tuesday, June 14, 2005

Group	x	y	z
A	3	9	12
B	9	21	30
C	5	9	14
Total	17	39	56

Temporary Variables

PROC REPORT handles temporary variables differently from report variables. Recall that these variables are initialized to missing (Step 2). Temporary variables are then retained throughout the procedure, only changing values from the execution of a COMPUTE block.

Returning to the first example, the following PROC REPORT identifies (and prints on the summary lines) the percentage of records in each group where the value of y is greater than 4.

```
proc report data=test nowd ;
  column Group x y n;
  define Group / order width=5;
  define x / display ;
  define y / display ;
  define z / computed ;
  define n / noprint ;

  break after Group / skip ;
  compute before group ;
    z=0 ;
    total=n ;
  endcomp ;
  compute y ;
    if y>4 then z+1 ;
  endcomp ;
  compute after Group ;
    yg=z/total ;
    line yg percent. 'of group ' Group $1. ' has y > 4' ;
  endcomp ;
run;
```

PROC REPORT follows the same steps as before; however, this example shows how the temporary variables, z, total, and yg, are handled.

1. **Create summary file.** This file has the additional column, n, which gives the number of nonmissing values.

Group	x	y	n
A	1	4	1
A	2	5	1
Summary for A	3	9	2
B	2	7	1
B	3	8	1
B	4	6	1
Summary for B	9	21	3
C	1	3	1
C	4	6	1
Summary for C	5	9	2
Overall Summary	17	39	7

2. **Initialize temporary variables.** These are z, total and yg.

Group	x	y	n	z	total	yg
	.	.	.	.	.	.

3. **Fill in values for first row.**
 - a. **Preliminary summary line.** This comes from the COMPUTE BEFORE Group block. Note that this is not written to the report, because there are no lines statements or BREAK with a SUMMARIZE option.

from temporary file

Group	x	y	n	z	total	yg
	3	9	2	.	.	.

from executing COMPUTE block

Group	x	y	n	z	total	yg
	3	9	2	0	2	.

- b. **Initialize first row.** Note that the values for z and total are retained.

Group	x	y	n	z	total	yg
	.	.	.	0	2	.

- c. **Fill in report variables from left to right.** This is a summary report, so only the summary lines are used.

values added from temporary file

Group	x	y	n	z	total	yg
A	1	4	1	0	2	.

from executing COMPUTE block for y (y is not greater than 4, so z does not change)

Group	x	y	n	z	total	yg
A	1	4	1	0	2	.

4. **Write the complete row to the report.** Note that only Group, x, and y are printed.

Steps 3.b, 3.c and 4 are repeated for the second row and produce the following. For this row z is incremented by 1, since y is greater than 4.

Group	x	y	n	z	total	yg
A	2	5	1	1	2	.

The next row to be produced is the first summary row. First, a preliminary summary line is produced using the temporary file and the retention of the temporary variable values.

Group	x	y	n	z	total	yg
A	.	.	2	1	2	.

At this point the COMPUTE block is executed, but because y is missing, z does not change. This is an important consideration, however, if an analysis type, such as SUM, is used. The COMPUTE AFTER Group block is then executed, computing the value of yg.

Group	x	y	n	z	total	yg
A	.	.	2	1	2	.50

Finally the LINE statement writes out to the report "50% of group A has y>4."

After the summary line is written, the process starts over again, with the COMPUTE BEFORE Group block being executed. This causes z and total to be reset to 0 and N, respectively, while yg will retain the previous value until changed when the COMPUTE AFTER Group block is executed.

The final report looks like this.

08:25 Thursday, June 16, 2005

Group	x	y
A	1	4
	2	5
50% of group A has y>4		
B	2	7
	3	8
	4	6
100% of group B has y>4		
C	1	3
	4	6
50% of group C has y>4		

Conclusion

Working with PROC REPORT can sometimes be frustrating when it is not clear what is going on “under the covers.” It is hoped the knowledge of the procedure’s sequence of steps will give users a better understanding of what is happening, allowing them to take advantage of the full power of the procedure.

The author would like to thank Dr. Sue Rakes, Regional Education Director in the Austin Office of SAS Institute, Inc., and Dan Hammarstrom, Systems Analyst at the Office of the Attorney General of Texas, Child Support Division, for valuable assistance in reviewing the paper and presented code. All errors and omissions are, of course, the fault of the author.

Contact information:

Keith Cranford

keith.cranford@cs.oag.state.tx.us