

PROC REPORT, A Guided Tour

Martin Haase

Department of Health Services,

School of Public Health and Community Medicine, University of Washington
& Medalia Healthcare LLC

Abstract

This paper demonstrates a process for rapid creation of a series of production reports using the REPORT procedure. You are first introduced to the REPORT procedure, and its key components. You then examine examples of code and the corresponding output which illustrate the features of the procedure. This is followed by a discussion of the limitations faced when using the REPORT procedure and some options for working around these limitations.

This discussion focuses on using PROC REPORT in a nonwindowing environment. The sample data set used for this paper replicates a health insurance claims data set.

What is PROC REPORT?

PROC REPORT is a 'report writer' intended to generate columnar output. As stated in SAS *Technical Report P-258 Using the REPORT Procedure in a Nonwindowing Environment, Release 6.07*, "The REPORT procedure combines features from the PRINT, MEANS, and TABULATE procedures with features of DATA step report writing into a powerful report-writing tool."

By replacing conventional code containing procedures such as SUMMARY and SORT, as well as multiple data steps, the samples used in this paper illustrate concise code which is much more easily adapted to new reporting needs.

Components of PROC REPORT

The REPORT procedure can be classified into six basic components: Procedure Line, Column Line, Define Statement, Compute Statement, Break Line, and Titles/Footnotes. The following describes these components along with the key options I most commonly use. Note that not all options available in the non-windowing REPORT procedure are listed.

1. The Procedure Line - In addition to invoking the procedure and specifying the data set, this line also establishes format and output options that are applied to the entire report.

Procedure Line Example

```
proc report data=mhos2.pragt headline headskip  
split='*'; (nowindows missing)
```

HEADLINE - This generates a line under all column headers and spaces on all pages of the report.

HEADSKIP - This option inserts a single blank line between the column headers, including the headline, and the data.

SPLIT - This option designates the split character in the column headers. The default value is '/', but if you use this character in the column headings, you can override it.

NOWINDOWS - With PC SAS, the output is diverted from the REPORTS window to the OUTPUT window.

MISSING - As with other SAS procedures, this groups missing values. For this example, the data contains no missing values. It is strongly recommended that this option be used, though.

2. Column Line - This line establishes the column order of variables in the report. This line begins with the word 'column' and is followed with the list of variables to be contained in the report. Listed variables that are not to be printed on the report are handled in their define statements. There are variables that can be used within compute statements that are not listed in the column statements. These variables are for internal computations within the REPORT procedure, such as deriving cumulative percents.

Column Line Example

```
column year hospital client tos tot_clms chg fee  
svcfec;
```

The order does matter. When executed, the variables will be analyzed and subsequently reported in order from left to right. This means that when a column is calculated, all columns included in the calculation must be listed before it.

3. Define Statement - For each variable listed in the column line, a corresponding define statement is required. The define statements do not have to be in column line order, however, the code remains manageable if in the same order. These define statements are the heart of the REPORT procedure. From these statements, the data are consolidated, summaries and statistics performed, and formats for column headers and data are established.

Define Statement Examples

```
define year / group format=4. width=4 center 'YEAR';
define tos / across format=$tos. width=10
'TYPE*SERVICE';
define chg / sum format=dollar12. width=12
'CHARGES';
define fee / noprint sum format=dollar12. width=12
'FEE';
define pctsvc / computed format=percent8.2 width=8
'PERCENT*SERVICE*FEE';
```

Sequence of the options to the right of the '/' does not affect execution of the statement. All definition statements must begin with 'define <variable>'. To keep the code efficient, I have standardized the position of each option included in a define statement.

NOPRINT - With this option, the column will not be printed. This is most commonly used for variables that are included in compute statements.

Operators - There are six possible options. They are ACROSS, ANALYSIS, COMPUTED, DISPLAY, GROUP, and ORDER. For each variable listed in the column line, only one of these operators can be selected.

ACROSS - This operator produces a column for each variable listed. The syntax for the across statement appears in the column and define statements.

Column Line -- Across Example

```
column year hospital tos, (tot_clms chg svcfee
pctsvc);
define tos / across format=$tos. width=10
'TYPE*SERVICE';
```

In the sample code, the 'tos' variable has two possibilities, INPATIENT or OUTPATIENT. The report will have a column for the four variables for each of the 'tos' values. The operation performed for each of the four variables is determined by the operator contained in each of their respective define statements. (See Figure 5.)

While the ACROSS operator creates an output that lends itself to a comparative report, I find that there are two drawbacks that limit its use. First, there is a physical space constraint. All variables included within the parentheses are repeated for each unique value contained in the across variable. Second, if calculated columns are included, the compute statements become extremely difficult to manage with column number references that do not lend themselves to being recycled. Adding or removing columns appearing before the across operator means that all column reference notation must be changed. Because of this, I do not recommend including the across operator unless absolutely necessary.

ANALYSIS - This operator executes a selected statistic (sum, mean, n, std, min, max, etc.) on a numeric variable. For this demonstration, sum is the only ANALYSIS function used. Note that the word 'ANALYSIS' is not required in the define statement. Only the statistic must be listed in the statement.

COMPUTED - With this operator, the column will contain a calculated value. The calculation is performed in the compute statement described below.

GROUP - This operator consolidates analysis and computed variables based on the values included in the group variable. If multiple variables are designated as GROUP, the grouping occurs from left to right, as indicated in the column line. By default, the GROUP operator orders rows contained within the group in ascending order. The group value will only be listed for the first occurrence within the group.

The DISPLAY and ORDER operators are not used here. The DISPLAY operator lists all values in a column, and is not used with these reports since it does not perform a summary function. The ORDER operator will order values, and its syntax and execution are very similar to the GROUP operator.

FORMAT - This assigns a format to the values to be contained in the column. All SAS formats are applicable.

WIDTH - This determines the number of spaces reserved for the length of the column. Make sure to allow enough room so that formatted values are not truncated.

CENTER (LEFT and RIGHT) - If desired, the column's contents can be center, left or right aligned.

TITLE - The last part of the define statement is a label for the column header. (The word 'title' is not used.)

4. Compute Statement - The compute statement provides the REPORT procedure with its flexibility in reporting. Within a compute statement, a wide variety of analysis operations can be performed. It is also possible to set up logic for conditional execution of the calculation.

Compute Statement Example

```
compute pctsvc;
    pctsvc=svcfec.sum/chg.sum;
endcomp;
```

Each routine begins with the word 'compute' followed by the variable listed in the column line and define statement. The routine ends with the word 'endcomp.' Within these two clauses a wide variety of choices are available. In the above example, a service fee to charge ratio is calculated. Note that the 'svcfec' and 'chg' variables are referenced as 'svcfec.sum' and 'chg.sum.' Since the 'svcfec' and 'chg' are summed (this is determined in the define statement for each of the variables), the original variables no longer exist but their summed values do.

I prefer to use compute statements for all new variables not otherwise included in the original data set. This does not have to be the case. For example, an average charge per claim could be calculated in the column line with the following:

```
column hospital year tot_clms chg (chg),mean;
```

However, by performing such an operation at the column line, a great deal of control is lost over the reported value. There is no ability to use conditional logic as could be done in a compute statement. It also reduces the flexibility of the report code in its ability to be recycled.

5. Break Line - The break line consists of the word 'break' followed by the variable to break after, a set of options for summarizing at a group level, controlling line titles, and inserting lines above and/or below the summaries. The variable included in a break line must have the GROUP operator in its define statement. Multiple break lines can be used, but each must have its own BREAK statement. With multiple break lines present, the break order will be reverse of how the group variables are listed in the column line. In other words, breaks will occur from right to left whenever a group variable changes value.

Break Line Examples

```
break after hospital / suppress summarize ol skip;
break after year / summarize ol skip;
rbreak after / suppress summarize dol dul skip;
```

AFTER - This indicates that the break line will occur after the last row in a group. (There is also a BEFORE option that places the summary data above the row data.)

SUPPRESS - This option suppresses a line title that indicates the variable being summarized.

SUMMARIZE - This summarizes all numeric variables. For an analysis variable, the same operation is performed over the entire group and reported. If a compute statement is used, the SUMMARIZE will perform the operation over the entire group.

OL, UL, DOL, DUL - There are four options for drawing lines above and below the summary values reported on the break line. They are: OL - over line (single), UL - underline (single), DOL - over line (double), DUL - under line (double).

SKIP - This option will insert a blank line before summarizing the numeric variables.

For the end of the report, there is the RBREAK statement. This RBREAK will report an overall summary for the entire report. With the exception of having no variable listed, this statement has the same syntax as the BREAK statement.

6. Titles and Footnotes - These are the titles and footnotes for the report.

Building a Report

This next section will use the components and options listed above to create a series of reports for profiling health insurance claims data. The data set used for these examples contains the following variables:

Variable	Format	Description
year	Numeric	Year claim was incurred
hospital	Character	Hospital submitting the claim
tos	Character	Type of service provided by the hospital. Raw data values are 'I' and 'O'. After format is applied values are INPATIENT and OUTPATIENT
client	Character	The Client paying the claim
tot_clms	Numeric	Number of claims
chg	Numeric	Charge per claim
fee	Numeric	Internal fee applied to each claim
svcfec	Numeric	Service fee paid by the client

For this series, I have classified the reports into utilization and distribution profiles. The utilization reports contain summarized charge and claims data, along with mean values and ratios. The distribution reports will expand on the utilization reports with percentage distributions between hospitals, clients, and years. Because of limited space, not all reports in the entire series are included. However, there are enough reports to see how the code is carried from one report to the next and expanded upon through each step.

1. Identify Objectives / Rough Sketch the Base Report

- Before beginning to develop the code, you must clearly know what questions the reports are trying to answer. With this in hand, try sketching out a couple of rough drafts of preliminary reports. Do not attempt to sketch out every possibility because the code is versatile enough to create literally dozens of reports. Sketch enough out to feel comfortable with the layout design.

2. The Define Statements - With Step 1 complete, write the define statements for each of the variables. This is done for two reasons. First, the nature of the define statements tends not to change from one report to next. For example, the 'chg' variable will be summed whether it is grouped by hospital, or by hospital and client. This also holds true for the character variables. Most likely, they will be used for grouping the dollar and claims variables. Second, by creating the define statements before assembling the report, I will have an approximation of line size before adding additional calculated fields. As is most often the case, the physical limits of the page restrict the number of columns that can be included in a report.

3. The First Report - I begin with the most basic report layout. This report contains variables only included in the original data set without any calculated fields or percentages. This is in Figure 1.

4. The Utilization Profiles - Now using the first report as a foundation, additional reports can be developed rapidly by changing how the data will be grouped. This is done by changing the column line order of the variables with GROUP operators and the corresponding BREAK statements. A sample of this is in Figure 2. This sample has two additional variables: 1) 'avgchg' is the mean charge per claim; 2) 'pctsvc' is the service fee to charge ratio. This ratio has replaced the actual service fee dollar value.

5. The Distribution Profiles - Expanding on the utilization reports, the distribution reports add

percentages for answering comparative questions regarding this data. Samples are in Figures 3 and 4. Figure 4 contains cumulative percentages for claims and charges. Lastly, I have included a report, Figure 5, that employs the ACROSS operator for comparing claims and dollars between inpatient and outpatient service. As noted earlier, because of the calculation notations required with this operator, I do not recommend including this in the main body of a report series unless absolutely necessary.

Limitations of PROC REPORT

There are four important limitations that can hinder the ability to produce all the reports you desire. The following is a brief highlight of these and my method for working around each of them. Of course, there are solutions to these limitations that I have not yet discovered.

1. Order/Ranking - As is often the case, a report that ranks hospitals or clients by total claims or charges is required. This type of report has proven to be rather difficult with the REPORT procedure because a variable that is summed cannot also be used for determining order. When this type of report is required, I must first summarize the data before executing PROC REPORT. In this case, I am using the procedure only for displaying the final results calculated in previous steps.

2. Control over the break statements - While the define statements and compute statements provide precise control of the data contained in the report, there is no equivalent method for controlling data in the BREAK statement. This is especially apparent with RBREAK. Notice that Figures 3 and 4 do not have overall summaries. The RBREAK will sum all numeric variables contained in the report. Therefore, percentages and cumulative percentages will be greater than 100%. I get around this by not using RBREAK and certain combinations of BREAK statements when dealing with these type of calculations.

3. Keeping a group together - When the data for a group variable has been summarized, the group summary (created with a BREAK) may print on the first line on top of the next page. This is due to the page size limitations, and there is no method for controlling when a break will span pages. To get around this, I strongly recommend reading Cynthia Stetz's "Controlling Page Breaks When Using PROC REPORT," *Proceedings of the Seventh Annual NorthEast SAS Users Group Conference, 1994*.

4. Paper only - While not necessarily a limitation, PROC REPORT produces only paper output. The summaries produced during execution of the procedure cannot be saved to a new data set. For cases when the summarized data needs to be saved, MEANS, SUMMARY, or other procedures must be used.

Conclusion

This paper has identified the key components required to build a REPORT procedure report and illustrated the versatility of the procedure to develop a simple report series.

References

SAS Institute, Inc. (1993), *SAS Technical Report P-258 Using the REPORT Procedure in a Nonwindowing Environment, Release 6.07*, Cary, NC: SAS Institute Inc.

Stetz, Cynthia. "Controlling Page Breaks When Using PROC REPORT," *Proceedings of the Seventh Annual NorthEast SAS Users Group Conference*, 1994. SAS Institute, Inc. (1994), Cary, NC: SAS Institute Inc.

Martin Haase
8818 NE 137th Street
Kirkland, WA 98034
(425) 823-7005
(206) 320-5236

e-mail:
mhaase@u.washington.edu
mahaase@ix.netcom.com

Figure 1

```
proc report data=mhos2.pragt headline headskip split='*';
column year hospital client tos tot_clms chg fee svcfee;
```

Report #1

```
define year / group format=4. width=4 center 'YEAR';
define hospital/ group format=$20. width=20 'HOSPITAL';
define client/ group format=$10. width=10 'CLIENT';
define tos/ group format=$tos. width=10 'TYPE*SERVICE';
define tot_clms/ sum format=comma9. width=9 'CLAIMS';
define chg/ sum format=dollar12. width=12 'CHARGES';
define fee/ sum format=dollar12. width=12 'FEE';
define svcfee/ sum format=dollar12. width=12 'SERVICE*FEE';
```

```
break after year / summarize ol skip;
break after hospital / suppress summarize ol skip;
rbreak after / suppress summarize dol dul skip;
```

title1 'Report #1';

YEAR	HOSPITAL	CLIENT	TYPE SERVICE	CLAIMS	CHARGES	FEE	SERVICE FEE
1992	HOSPITAL #1	CLIENT #1	INPATIENT	6	\$131,527	\$2,389	\$129,137
			OUTPATIENT	51	\$46,396	\$1,356	\$45,040
		CLIENT #2	INPATIENT	16	\$156,533	\$3,206	\$153,327
			OUTPATIENT	77	\$44,790	\$1,287	\$43,502
		CLIENT #3	INPATIENT	7	\$25,803	\$722	\$25,081
			OUTPATIENT	58	\$55,284	\$1,400	\$53,884
				225	\$460,333	\$10,361	\$449,972
-----	-----	-----	-----	-----	-----	-----	-----
1992				2,247	\$4,662,234	\$106,489	\$4,555,745
=====	=====	=====	=====	=====	=====	=====	=====
				10,000	\$23,806,097	\$545,148	\$23,260,036
=====	=====	=====	=====	=====	=====	=====	=====

Figure 2

```
proc report data=mhos2.pragt headline headskip split='*';
column hospital tos year tot_clms chg avgchg fee svcfee pctsvc;
```

Report #2

```
define hospital/ group format=$20. width=20 'HOSPITAL';
define tos/ group format=$tos. width=10 'TYPE*SERVICE';
define year / group format=4. width=4 center 'YEAR';
define tot_clms/ sum format=comma9. width=9 'CLAIMS';
define chg/ sum format=dollar12. width=12 'CHARGES';
define avgchg / computed format=dollar9. width=9 'AVG*CHARGE';
define fee/ sum format=dollar12. width=12 'FEE';
define svcfee/ noprint sum format=dollar12. width=12 'SERVICE*FEE';
define pctsvc/ computed format=percent8.2 width=8
'PERCENT*SERVICE*FEE';
```

```
compute avgchg; avgchg=chg.sum/tot_clms.sum; endcomp;
compute pctsvc; pctsvc=svcfee.sum/chg.sum; endcomp;
```

```
break after tos / suppress summarize ol skip;
break after hospital / suppress summarize ol skip;
rbreak after / suppress summarize dol dul skip;
title1 'Report #2';
```

HOSPITAL	TYPE SERVICE	YEAR	CLAIMS	CHARGES	AVG CHARGE	FEE	SERVICE FEE
HOSPITAL #1	INPATIENT	1992	29	\$313,863	\$10,823	\$6,317	97.99%
		1993	34	\$355,687	\$10,461	\$5,652	98.41%
		1994	59	\$820,556	\$13,908	\$16,622	97.97%
			122	\$1,490,106	\$12,214	\$28,591	98.08%
	OUTPATIENT	1992	196	\$146,470	\$747	\$4,043	97.24%
		1993	324	\$270,308	\$834	\$7,089	97.38%
		1994	357	\$414,661	\$1,162	\$10,598	97.44%
			877	\$831,440	\$948	\$21,730	97.39%
			999	\$2,321,545	\$2,324	\$50,321	97.83%
=====	=====	=====	=====	=====	=====	=====	=====
			10,000	\$23,806,897	\$2,381	\$545,148	97.71%
=====	=====	=====	=====	=====	=====	=====	=====

Figure 3

```
proc report data=mhos2.pragt headline headskip split='*';
column year hospital tot_clms pctclms chg pctchg;
```

Report #3

```
define year / group format=4. width=4 center 'YEAR';
define hospital/ group format=$20. width=20 'HOSPITAL';
define tot_clms/ sum format=comma9. width=9 'CLAIMS';
define chg/ sum format=dollar12. width=12 'CHARGES';
define pctclms / computed format=percent8.1 width=8
'PERCENT*ANNUAL*CLAIMS';
define pctchg / computed format=percent8.1 width=8
'PERCENT*ANNUAL*CHARGES';
```

```
* 'compute before year' to generates internal summary for denominators;
compute before year; anl_clms=tot_clms.sum; anl_chg=chg.sum; endcomp;
compute pctclms; pctclms=tot_clms.sum/anl_clms; endcomp;
compute pctchg; pctchg=chg.sum/anl_chg; endcomp;
```

```
break after year / suppress summarize ol skip;
title1 'Report #3';
```

YEAR	HOSPITAL	CLAIMS	PERCENT ANNUAL CLAIMS	CHARGES	PERCENT ANNUAL CHARGES
1992	HOSPITAL #1	225	10.0%	\$460,333	9.9%
	HOSPITAL #2	940	41.8%	\$2,092,024	44.9%
	HOSPITAL #3	534	23.8%	\$1,268,158	27.2%
	HOSPITAL #4	549	24.4%	\$843,478	18.1%
		2,248	100.0%	\$4,663,993	100.0%

Figure 4

```
proc report data=mhos2.pragt headline headskip split='*';
column client hospital tot_clms pctclms cumclms chg pctchg cumchg;
```

```
define client/ group format=$10. Width=10 'CLIENT';
define hospital/ group format=$20. Width=20 'HOSPITAL';
define tot_clms/ sum format=comma9. Width=9 'CLAIMS';
define chg/ sum format=dollar12. Width=12 'CHARGES';
define pctclms / computed format=percent8.1 width=8
'PERCENT*CLIENT*CLAIMS';
define cumclms / computed format=percent8.1 width=8
'CUM*PERCENT*CLIENT*CLAIMS';
define pctchg / computed format=percent8.1 width=8
'PERCENT*CLIENT*CHARGES';
define cumchg / computed format=percent8.1 width=8
'CUM*PERCENT*CLIENT*CHARGES';
```

```
compute before client; * denominators for each client group;
    anl_clms=tot_clms.sum;
    clm0=0; * reset to 0 with each client group;
    anl_chg=chg.sum;
    chg0=0;
endcomp;

* client percentages;
compute pctclms; pctclms=tot_clms.sum/anl_clms; if hospital ne '' then
clm0+pctclms; endcomp;

compute pctchg; pctchg=chg.sum/anl_chg; if hospital ne '' then
chg0+pctchg;
endcomp;

* cumulative percentage;
compute cumclms;
    cumclms=clm0;
    if hospital = '' then cumclms=''; * no summary of cumulative
    percents;
endcomp;

compute cumchg; cumchg=chg0; if hospital = '' then cumchg=''; endcomp;

break after client / suppress summarize ol skip;
title1 'Report #4';
```

Report #4

CLIENT	HOSPITAL	CLAIMS	PERCENT	CUM	CHARGES	PERCENT	CUM
			CLIENT	CLIENT		CLIENT	CLIENT

CLIENT #1	HOSPITAL #1	281	10.8%	10.8%	\$719,327	12.9%	12.9%
	HOSPITAL #2	995	38.4%	49.2%	\$2,413,652	43.4%	56.3%
	HOSPITAL #3	663	25.6%	74.8%	\$1,263,163	22.7%	79.0%
	HOSPITAL #4	653	25.2%	100.0%	\$1,166,544	21.0%	100.0%
			2,592	100.0%			
					-----	-----	-----
					\$5,562,686	100.0%	

Figure 5

```
proc report data=mhos2.pragt headline headskip split='*';
column year hospital tos, (tot_clms chg svcfee pctsvc);
```

```
define year / group format=4. width=4 center 'YEAR';
define hospital/ group format=$20. Width=20 'HOSPITAL';
define tos/ across format=$tos. width=10 'TYPE*SERVICE';
define tot_clms/ sum format=comma9. Width=9 'CLAIMS';
define chg/ sum format=dollar12. width=12 'CHARGES';
define svcfee/ noprint sum format=dollar12. width=12 'SERVICE*FEE';
define pctsvc/ computed format=percent8.2 width=8
'PERCENT*SERVICE*FEE';
```

```
compute pctsvc;
    _c6=_c5/_c4_;
    _c10=_c9/_c8_;
endcomp;
* NOTE -- addition or removal of columns before the across operator
will invalidate these calculations. nw column references
will be required.;
```

```
break after year / suppress summarize ol skip;
rbreak after / suppress summarize dol dul skip;
title1 'Report #5';
```

Report #5

YEAR	HOSPITAL	CLAIMS	TYPE SERVICE					
			INPATIENT		PERCENT SERVICE FEE	OUTPATIENT		PERCENT SERVICE FEE
			CHARGES			CLAIMS	CHARGES	
1992	HOSPITAL #1	29	\$313,863	97.99%	196	\$146,470	97.24%	
	HOSPITAL #2	101	\$1,440,050	97.89%	839	\$651,974	97.38%	
	HOSPITAL #3	48	\$912,236	97.77%	486	\$355,922	97.38%	
	HOSPITAL #4	58	\$460,882	97.98%	491	\$382,597	97.45%	
		236	\$3,127,031	97.88%	2,012	\$1,536,962	97.39%	