
Technical Paper

The REPORT Procedure: Getting Started
with the Basics

Table of Contents

Introduction	1
REPORT Procedure: Syntax and General Description	1
COLUMN Statement	3
Spanning a Text String across Column Headers	3
Using an Across Variable in a COLUMN Statement	4
Using an Alias for a Column Variable Name	5
DEFINE Statement	5
DEFINE Statement Options for Displaying and Using Variables	5
DEFINE Statement Options for Customizing Reports	9
Additional PROC REPORT Statements	10
BY Statement.....	10
BREAK Statement	11
COMPUTE Statement	12
Compute Block Associated with a COLUMN Statement Variable	13
Compute Block Associated with a Target Location	14
Common Compute-Block Issues	15
FREQ Statement	15
RBREAK Statement.....	16
WEIGHT Statement	16
References	17

Introduction

The REPORT procedure (PROC REPORT) combines features of the PRINT, TABULATE, and MEANS procedures, along with features of the DATA step, in a single report-writing tool that can be used to produce a variety of reports. You can use PROC REPORT in both windowing and non-windowing environments; however, this paper focuses on using PROC REPORT in a non-windowing environment. This paper introduces PROC REPORT syntax and functionality to audiences with little or no experience with PROC REPORT.

REPORT Procedure: Syntax and General Description

PROC REPORT supplies raw data and style definitions that enable the SAS Output Delivery System (ODS) to display output in various formats.

Syntax for PROC REPORT:

```

PROC REPORT <option(s)>;
  BREAK location break-variable</ option(s)>;
  BY <DESCENDING> variable-1 <...><DESCENDING> variable-n <NOTSORTED>;
  COLUMN column-specification(s);
  COMPUTE location <target> </ STYLE=<style-element-name> <[style-attribute-specification(s)]>>;
    LINE specification(s);
    ... select SAS language elements ...
  ENDCOMP;
  COMPUTE report-item </ type-specification>;
    CALL DEFINE (column-id, 'attribute-name', value);
    ... select SAS language elements ...
  ENDCOMP;
  DEFINE report-item / <usage> <attribute(s)> <option(s)> <justification> <COLOR=color>
    <'column-header-1' <...>'column-header-n'>> <style>;
  FREQ variable;
  RBREAK location </ option(s)>;
  WEIGHT variable;
RUN;

```

In this syntax, the PROC REPORT statement is the only required statement. Therefore, the most basic PROC REPORT program is as follows:

```

proc report data=sashelp.class nowd;
run;

```

Output

Name	Sex	Age	Height	Weight
Alfred	M	14	69	112.5
Alice	F	13	56.5	84
Barbara	F	13	65.3	98
Carol	F	14	62.8	102.5
Henry	M	14	63.5	102.5

If you do not specify a data set name, the processor will pick up the latest data set name. It also picks up the LINESIZE=, PAGESIZE=, and CENTER/NOCENTER system options as well as the default split character (/). The options you use in the PROC REPORT statement apply to the entire report in both the listing and the ODS destinations.

This output is similar to output created by using the OBS= option in the PRINT procedure. This particular PROC REPORT example looks simple enough, but other functions are taking place behind the scenes. For example, if you include the LIST option in the PROC REPORT statement, you will see an expanded version of the code that shows additional statements, as shown in this example:

```
proc report data=sashelp.class ls=78 ps=60 split="/" center;
  column Name Sex Age Height Weight;
  define Name / display format=$8. width=8 spacing=2 left "Name";
  define Sex / display format=$1. width=1 spacing=2 left "Sex";
  define Age / sum format=best9. width=9 spacing=2 right "Age";
  define Height / sum format=best9. width=9 spacing=2 right "Height";
  define Weight / sum format=best9. width=9 spacing=2 right "Weight";
run;
```

The LIST option is a handy debugging option. It is great for solving PROC REPORT logic problems, especially when the code is written with SAS[®] macros. The LIST option produces the shell of what PROC REPORT is using to build the report in memory. The option **does not** currently include global statements, the compute block (see ["COMPUTE Statement"](#)) for the _PAGE_ target, ODS style elements, and the NOWINDOWS option.

The following list describes some of the most common options used in the PROC REPORT statement:

- **HEADLINE option**—underlines all of the column headers (and spaces that are between the headers) at the top of each page of your report. This option only applies when you list output. For formatted output in ODS, you need to use a style element to create a headline effect.
- **HEADSKIP option**—inserts a blank line beneath the column headers. If you use the HEADLINE option as well, the HEADSKIP option inserts a blank line beneath the underline created by the HEADLINE option. Like the HEADLINE option, the HEADSKIP option only applies when you list output. For formatted output in ODS, you need to use a style element to create a headline effect.
- **MISSING option**—considers missing values for any group, order, or across variable as valid. This option does not create all possible combinations of a value if a combination does not occur in the data. In such cases, you can use the PRELOADFMT option in the DEFINE statement instead. To apply the MISSING option only to a particular group, order, or across variable, add the option in the DEFINE statement instead of in the PROC REPORT statement.
- **NOWINDOWS | NOWD option**—specifies that the code will be run in a non-windowing environment. By default, if you **do not** specify this option, the report will run in windowing mode and the output will show up in the REPORT window. ODS output is NOT created in windowing mode.

- **SPLIT= option**—specifies the split character. The split character applies to the data and the header in the listing, and it usually applies to the header in the ODS destinations. The default split character is the forward slash (/). If you use a forward slash in the text of your report and you use in-line formatting, you will need to change the split character to a character that is not in the report. Text. Generally, the caret (^) or the tilde (~) characters are safe split characters to use if your report is in English. However, if your report is in a language that uses the caret or tilde as diacritical marks and these characters are in the text of your report, you will then need to choose a different character.
- **STYLE option**—specifies one or more style elements to be used in various locations of a report when you use ODS. The syntax for the STYLE option is as follows:

```
STYLE<(location(s))=<style-element-name><[style-attribute-specification(s)]>
```

Location values for the STYLE option are CALLDEF, COLUMN, HEADER, LINES, REPORT, and SUMMARY. Which of these values are valid and default depends on what statement the STYLE option appears in. For *style-element-name* and *style-attribute-specification* values, see the following section in SAS 9.1.3 OnlineDoc:

SAS Procedures ► REPORT ► Concepts: REPORT Procedure ► Using Style Elements in PROC REPORT

The following sections explain each of the PROC REPORT statements in more detail and provide examples to illustrate the concepts.

COLUMN Statement

The COLUMN statement enables you to define the arrangement of columns and headers that span more than one column. The COLUMN statement is similar to the VAR statement in the PRINT procedure in that the data set variables, statistics, and computed variables are listed in the order of appearance, left to right (and top to bottom), in the report. If, as shown in the following example, a COMPUTE statement block for a variable references another variable that is positioned to the right in the COLUMN statement, the result will be incorrect or missing.

```
proc report nowd data=sashelp.class;
  col name newage age;
  compute newage;
    newage=age.sum*100;
  endcomp;
run;
```

The following sections explain several items to keep in mind when you use the COLUMN statement.

Spanning a Text String across Column Headers

In the COLUMN statement, you can span a text string across column variables. The text string must be placed in single quotation marks, and the text string as well as the variables that are to be spanned must be enclosed in parentheses. The following example shows how to create three column variables for a report and how to span a text string across two of those column headers.

```
proc report nowd data=sashelp.class headline;
  col ('Span over variables' var1 var2) var3;
run;
```

Output

```
Span over variables
  var1      var2      var3
```

If you forget the quotation marks or do not place the appropriate variables inside the parentheses, you will get incorrect output, as shown in the following two examples.

If you forget the ending quotation mark, the output will look like this:

```
Span over variables var1 var2 var3
```

If you forget to place all of the variables that you want spanned inside the parentheses, the text string will only span the variable or variables that are inside the parentheses. For example, suppose you want the text to span var1 and var2, but you place var2 outside of the parentheses:

```
proc report nowd data=sashelp.class headline;
  col ('Span over variables' var1) var2 var3;
run;
```

In this case, the output looks like this:

```
Span over variables
      var1      var2      var3
```

Note: Be aware that if the first and last characters of your spanning text string are certain special characters (- . = \ _ * : +), PROC REPORT will expand the header by repeating the character before and after the text string in the output, as shown in the following example:

```
proc report nowd data=sashelp.class;
  col ('*span*' name age);
run;
```

Output

```
*****span*****
Name      Age
Alfred    14
Alice     13
```

This behavior occurs only with list output.

Using an Across Variable in a COLUMN Statement

An *across variable* is a variable for which PROC REPORT creates a column for each value of that variable. In a COLUMN statement that contains an across variable, a comma is required between the across variable and the variables that are to be displayed beneath the across variable. In the following COLUMN statement, VAR2 is the across variable, which has been defined earlier in the associated program in a DEFINE statement.

```
col var1 var2,(n var3);
```

- If you do not request a variable or a statistic with the across variable, each across column will default to the N statistic. This behavior causes the default N statistic to be stacked underneath.
- If the comma is missing, then no stacking of data occurs.

Using an Alias for a Column Variable Name

You can use an alias for any column variable name that will be used in more than one column in your report. An alias helps to distinguish the items that have the same variable name. The syntax for assigning an alias is *report-item=name*, where *report-item* is the column variable name and *name* is the alias. In the following example, AGE is the column variable name and NEWAGE is the alias.

```
proc report nowd data=sashelp.class list;
  col age age=newage age;
run;
```

Output (produced from the LIST option in the PROC REPORT statement)

```
PROC REPORT DATA=SASHELP.CLASS LS=181 PS=84 SPLIT='/' CENTER;
COLUMN   (Age Age=newage Age=_A1);

DEFINE   Age / SUM FORMAT= BEST9. WIDTH=9   SPACING=2   RIGHT "Age" ;
DEFINE   newage / SUM FORMAT= BEST9. WIDTH=9   SPACING=2   RIGHT "Age" ;
DEFINE   _A1 / SUM FORMAT= BEST9. WIDTH=9   SPACING=2   RIGHT "Age" ;
RUN;
```

If you do not assign an alias, PROC REPORT assigns one, and it appears in the list output.

DEFINE Statement

After you set your column variables, the next step is to use the DEFINE statement to describe how PROC REPORT should use and display each report item (column variable, in this case).

Each variable in the COLUMN statement should have a separate DEFINE statement. If a DEFINE statement is not included for a variable, PROC REPORT will use a default DEFINE statement. For variables from the input data set, the default option for character variables in the DEFINE statement is DISPLAY; the default option for numeric variables is SUM. Statistics do not have a definition. Computed variables are numeric by default, unless a character format is used. Any formats and labels for the variables are passed from the input data set, as well.

DEFINE Statement Options for Displaying and Using Variables

The DEFINE statement has six options you can use that direct PROC REPORT on how to display or use variables from the input data set that appear in the COLUMN statement.

- **ACROSS option**—defines the data set variable as an across variable and creates a column for each value of the variable. This option also designates each value of the across variable as a header.

```
proc report nowd data=sashelp.class nocenter headline;
  col sex,(weight height);
  define sex / across;
  define weight / sum;
  define height / sum;
run;
```

Output

Sex			
F		M	
Weight	Height	Weight	Height
811	545.3	1089.5	639.1

Note: In a compute block, any column that is created by the ACROSS option is referred to by the column number in the form of `_Cn_`, where *n* is the column number. This behavior includes the NOPRINT variables, as shown in this example:

```
proc report nowd data=sashelp.class list;
  col age height sex,(weight myvar);
  define sex / across;
  define height / noprint;
  define age / group;
  compute myvar;
    _c4_=height.sum*10;
    _c6_=height.sum*10;
  endcomp;
run;
```

- **ANALYSIS option**—defines the data set variable as an analysis variable, which must be a numeric variable for any analysis to occur. The default statistic for analysis variables is sum, but you can use any valid statistic (for example, N, MEAN, MAX, MEDIAN, and MIN). If all the variables are numeric, the resulting output will be a summarized row, as shown in this example:

```
proc report nowd data=sashelp.class(obs=5) nocenter headline;
  col age weight;
  define age / sum;
  define weight / sum;
run;
```

Output

<u>Age</u>	<u>Weight</u>
61	394

Note: When an analysis variable is referred to in a compute block, you must use the associated statistic as well. For example, if you use the sum statistic for AGE, as shown in the following example, then you have to use AGE.SUM in the compute block.

```
proc report nowd data=sashelp.class;
  col age newage;
  define age / sum;
  define newage / computed;
  compute newage;
    newage=age.sum*100;
  endcomp;
run;
```

- **DISPLAY option**—defines data set variables as display variables and prints each observation in the order they appear in the input data set. The DISPLAY option can be applied to both numeric and character variables. The following example illustrates the use of the DISPLAY option for the variables Name, Sex, Age, and Weight, and it shows the output created by PROC REPORT:

```
proc report nowd data=sashelp.class(obs=5)nocenter headline;
  col name sex age weight;
  define name / display;
  define sex / display width=3;
  define age / display;
  define weight / display;
run;
```

Output

Name	Sex	Age	Weight
Joyce	F	11	50.5
Jane	F	12	84.5
Louise	F	12	77
Alice	F	13	84
Barbara	F	13	98

- **GROUP option**—defines the data set variable as a group variable and enables you to use a format to create groups. You apply the format by using the FORMAT= option in the DEFINE statement or by using a FORMAT statement. In the program, the format is applied first, before PROC REPORT creates the groups. All like-formatted values are grouped together, and the observations are consolidated. The key is to have all of the character variables defined as group variables; otherwise, the observations will not be consolidated.

```
proc report nowd data=sashelp.class nocenter headline;
  col sex age weight;
  define sex / group;
  define age / group;
  define height / sum;
run;
```

Output

Sex	Age	Weight
F	11	50.5
	12	161.5
	13	182
	14	192.5
	15	192.5
M	11	85
	12	310.5
	13	84

14	215
15	245
16	150

- Typically, a group variable is only printed (in the first row of the group) when the value changes (as shown in the output for the previous example). However, if there are other group or order variables to the left of that particular group variable, then the value for that variable will be repeated. For example, in the following output, the value (F) for the group variable SEX (F) is repeated because the group variable AGE appears to the left of SEX, and the values for AGE change. So the value for SEX is repeated on each line.

```
proc report nowd data=sashelp.class(obs=5);
  col age sex weight;
  define age / group;
  define sex / group;
run;
```

Output

```
S
e
Age x   Weight
13 F    182
14 F    102.5
M       216
```

- **ORDER option**—defines the data set variable as an order variable and enables you to order the rows in a report. You apply the format by using the FORMAT= option in the DEFINE statement or by using a FORMAT statement. In the program, the format is applied first, before PROC REPORT creates the groups.

```
proc report nowd data=sashelp.class;
  col sex age weight;
  define age / order;
  define sex / order;
run;
```

Output

```
Sex      Age      Weight
F        11        50.5
         12        84.5
         77
         13        84
         98
         14       102.5
         90
         15       112.5
         112
M        11        85
         12        83
         99.5
         128
         13        84
         14       112.5
         102.5
```

15	133
	112
16	150

- The values for the two order variables (SEX and AGE) are not repeated from row to row if the value does not change, unless a group variable to its left changes values. If that happens, the value for the order variable will be printed on that row.

DEFINE Statement Options for Customizing Reports

The DEFINE statement has a number of other options you can use to customize the appearance of your report. This section describes several of the most useful of these options. (For details on these and other options available in the DEFINE statement, see **SAS Procedures ► REPORT ► DEFINE Statement** in SAS 9.1.3 OnlineDoc.)

- **COMPUTED option**—defines a report item as a *computed variable*, which is a variable that you define specifically for your report. These variables are created by external code and are not included in the input data set. By default, the computed variable is numeric. A computed character variable requires a character format and the option /CHAR in the COMPUTE statement. A computed variable requires a compute block to obtain its value.
- **FLOW option**—wraps the value of a character variable in the column, based on the width of the column in the listing output. This option honors the split character in the listing output, but it will try to split text at a blank if no split character is designated. To achieve the appearance of flow in ODS output, you need to use the [STYLE<\(location\(s\)\)>= option](#), which controls the cell width.
- **FORMAT= option**—assigns a SAS or user-defined format. Keep in mind that the format is applied first to a group, order, or across variable prior to grouping like values. If you do not want this behavior, you can use a CALL DEFINE statement in a compute block as an alternative. The CALL DEFINE statement enables you to change the format of a cell rather than the format of the entire column (as the FORMAT= option does).
- **NOPRINT option**—is used to keep the column from printing in the report. The column is created in memory, so the values can still be used in any calculations. A NOPRINT variable column is counted when determining the column number for an across column in a compute block.
- **NOZERO option**—suppresses the display of a variable if its values are all zero or missing.
- **ORDER= option**—specifies how the group, order, or across variables should be ordered (sorted). The default value for this option is FORMATTED, which can be a problem for a date value because the sort order will be based on the formatted text rather than on the internal SAS date. For a date value, you can use ORDER=INTERNAL, which provides the same sort order you would get if you use the SORT procedure. Users sometimes confuse the ORDER = option with the ORDER option (defined in the previous section), but they are separate options that perform different tasks.
- **PAGE option**—inserts a page break just before printing the column that contains the PAGE option.
- **STYLE<(location(s))>= option**—specifies a style element for either column cells, column headers, or both locations. The syntax for specifying both a column header and a column cell is as follows:

```
STYLE(header_column)=<style-element-name>[<style-attribute-specification(s)>]
```

Additional PROC REPORT Statements

In addition to the COLUMN and DEFINE statements, PROC REPORT has several other statements that help you design your report. These statements are BY, BREAK, RBREAK, FREQ, WEIGHT, and COMPUTE.

BY Statement

The BY statement is used to create a separate report on individual pages for each BY group. If you request a default summary row (using the RBREAK statement) before or after the BY group, PROC REPORT creates a summary for each BY group rather than for the whole report.

In the following example, SAS will create individual report pages for the BY variable, SEX:

```
proc report nowd data=sashelp.class nocenter noheader;
  by sex;
  col age height;
  define age / group;
  define height / sum;
run
```

If you want the report total rather than BY groups on individual pages, modify the REPORT procedure as follows:

- Remove the BY statement and add the BY variables to the beginning of the COLUMN statement.
- Define the variables (either as group, order, or noprint variables) in DEFINE statements.
Note: A BY variable must be sorted outside of PROC REPORT. However, PROC REPORT sorts group, order, and across variables within the procedure.
- Create the BY line using the COMPUTE and LINE statements.

These modifications are shown in the following example:

```
proc report nowd data=sashelp.class Nocenter noheader;
  col sex age height;
  define sex / group noprint;
  define age / group;
  define height / sum;
  compute before _page_;
    line @1 'sex=' sex $1.;
    line ' ';
  endcomp;
  break after sex / page;
run;
```

Output

SEX=F

<u>Age</u>	<u>Height</u>
11	51.3
12	116.1
13	121.8
14	127.1
15	129

SEX=M

<u>Age</u>	<u>Height</u>
11	57.5
12	181.1
13	62.5
14	132.5
15	133.5
16	72

BREAK Statement

The BREAK statement produces a default summary row when the value of a GROUP or an ORDER variable changes. This statement includes the location for the break (BEFORE or AFTER) followed by a forward slash (/) and any options that you want to use. The following example indicates that you want to break after the Sex variable, and you want to start a new page after the last break line.

```
break after sex / page;
```

Other commonly used BREAK options:

- **SUMMARIZE option**—writes a summary line that totals all analysis variables.
- **STYLE option**—enables you to add style attributes
- **PAGE option**—forces a new page when the group or order variable changes values.
- **SUPPRESS option**—keeps the group and order variables from printing on the summary row.

The following formatting options are available for listing output only:

- **DOL option**—writes a double line **over** each value that appears in the summary line (or that would appear in the summary line if you were to specify the SUMMARIZE option). The default underline character used to designate a double overline is the equal (=) sign.
- **OL option**—writes a single line **over** each value that appears in the summary line (or that would appear in the summary line if you were to specify the SUMMARIZE option). The default overline character used to designate a single line is the hyphen (-).
- **DUL option**—writes a double line **under** each value that appears in the summary line (or that would appear in the summary line if you were to specify the SUMMARIZE option). The default double overline character used to designate a double overline is the equal (=) sign.

- **UL option**—writes a single line **under** each value that appears in the summary line (or that would appear in the summary line if you were to specify the SUMMARIZE option). The default overline character is the hyphen (-).
- **SKIP option**—adds a blank line as the last break line.

Notes

- These formatting options for listing output are not honored in ODS HTML output (or other ODS output). SAS Note 23671 provides a table with example code that can simulate these options in ODS output.
- In ODS, a group or order variable value will only repeat on a new page if the page formatting is defined with the PAGE option.

COMPUTE Statement

The COMPUTE statement starts a *compute block*, which contains one or more programming statements that PROC REPORT executes as it builds the report. A compute block can be associated with a variable in the COLUMN statement or with the target location (BEFORE or AFTER). The ENDCOMP statement ends the compute block.

The following example shows both types of compute blocks:

```
proc report nowd data=sashelp.class nocenter headline;
  col age sex, (n mypct);
  define age / group;
  define sex / across 'gender';
  define n / noprint;
  define mypct / computed format=percent8. '%' center;
  compute before;
    total1=_c2_;
    total2=_c4_;
  endcomp;
  compute mypct;
    _c3_=_c2_/total1;
    _c5_=_c4_/total2;
  endcomp;
run;
```

Output

Age	Gender	
	F	M
	%	%
11	11%	10%
12	22%	30%
13	22%	10%
14	22%	20%
15	22%	20%
16	.	10%

Compute blocks use the DATA step for processing. You can use any DATA step element (DO loops, arrays, %INCLUDE statements, all functions, and IF-THEN/ELSE statements) in the compute block.

Compute Block Associated with a COLUMN Statement Variable

As mentioned previously, the compute block can be associated with a COLUMN statement variable. The COLUMN statement variable that is associated with the compute block can be a data set variable, a computed variable, or a statistic. When you use a compute block with one of these variables, keep the following points in mind:

- Any variable that is located to the right of the compute-block variable will be considered missing, per the left-to-right rule discussed in the section [COLUMN Statement](#). In this case, either move the variable so that it is to the left of the compute block variable in the COLUMN statement or change the compute block so that it uses the variable that is to the right of the compute block variable. Also note that it is not necessary to have a separate COMPUTE block for every variable value that needs to be modified.
- An IF statement looks at the report rather than the data for evaluation. This is important to remember when you are evaluating group or order variables. A blank cell is evaluated as a blank value. In this case, create a DATA step variable in a COMPUTE BEFORE statement block to hold the value. Use the created DATA step variable in the evaluation instead of the group or order variable.

- A computed character variable requires a DEFINE statement with a character format, as shown in this example:

```
define dummy / computed format=$8.;
```

- A computed character variable requires a forward slash (/) and the CHAR option after the variable name in the COMPUTE statement:

```
compute dummy / char;
```

- You can use a CALL DEFINE statement to change the attribute of a cell, a column (`_COL_`), or a row (`_ROW_ONLY` in ODS):

```
call define(column-id,'attribute-name',value);
```

- A CALL DEFINE statement can change the format of a cell in the listing and the ODS destinations, as illustrated in this example:

```
call define('_c1_', 'FORMAT', 'DOLLAR8.2');
```

- For ODS, the most popular attributes in the CALL DEFINE statement are the STYLE= attribute and the URL attribute. The following example illustrates the use of the STYLE= attribute:

```
call define('_C1_', 'Style', 'style=[background=red]');
```

- Incorrect values for the style specifications in a CALL DEFINE statement might result in this error:

```
ERROR: Expecting a (. Syntax error, statement will be ignored.
```

- The `_BREAK_` automatic variable is used to determine what row (either the detail or the summary row) is being processed. If the row is a summary row, `_BREAK_` will contain the variable name that created either the break or the default summary row break created with `_RBREAK_`. The value for `_BREAK_` is blank for a detail row.

Compute Block Associated with a Target Location

The COMPUTE statement can be associated with a target:

```
COMPUTE location <target>
```

The compute block requires a location with an optional target. When you use a compute block with a target, keep the following points in mind:

- A LINE statement is used to print data at the location and target of the compute block. The LINE statement is similar to a PUT statement, but it will not print in the log. In ODS, a LINE statement is treated as one string.
- Values for *location* are BEFORE or AFTER. These values indicate where the compute block is executed in relation to the target. The value for *target* can be either blank (no value), `_PAGE_`, or a group or order variable.
 - A compute block that executes BEFORE the target processes prior to the detail data on the first page. The LINE statement text is placed after the headers.
 - A compute block that executes AFTER the target processes after the detail data on the last page. The LINE statement text is placed immediately after the detail data on the last page.
 - A compute block that executes BEFORE a target of `_PAGE_` processes prior to the column headers on every page. If a COLUMN statement variable is listed in the LINE statement, the value comes from the first detail row of the page.
 - A compute block that executes AFTER a target of `_PAGE_` processes after the detail data on every page prior to the footnote. If a COLUMN statement variable is listed in the LINE statement, the value comes from the last detail row of the page.
 - A compute block that executes before a group or order variable processes prior to the printing of a group or order value. The line statement will appear there as well.
 - A compute block that executes AFTER a group or order variable processes after the printing of a group or order value. The line statement will appear there as well.
- To add a style for text that is created by the LINE statement, set the STYLE= option after a forward slash (/) in the COMPUTE statement. The syntax for the STYLE= option is as follows:

```
COMPUTE location <target> </ STYLE=<style-element-name> <[style-attribute-specification(s)]>>;
```

Example:

```
compute before / style=[background=pink];
```

- You can add inline formatting to LINE statements, as shown in this example:

```
line `^S={foreground=green}GREEN^S={ }means go` ;
```

A LINE statement cannot be conditionally executed; it will **ALWAYS** print out. However, you can control what text in the LINE statement is printed by conditionally creating a value in a DATA step variable that is used in the LINE statement. The LINE statement will still be printed, but what is printed is the conditional text you defined in the DATA step variable. You can create DATA step variables at any time in a compute block. The values change only when they are coded to do so. Do NOT include a DATA step variable in the COLUMN statement. If you do, the value will reinitialize at every row rather than being retained across the rows.

- Character variables in a LINE statement require a format. Analysis variables in a LINE statement require a compound name in the form *variable-name.statistic*, but the format is optional. The default statistic is SUM.
- In listing output, a numeric group or order variable is set to blank internally, and it cannot be modified at the break created by an RBREAK statement. However, you can use one of the following methods to modify the value of the COMPUTED variable:
 - Define a computed variable (using the COMPUTED option in a DEFINE statement) to contain the value of the group or the order variable.
 - In ODS, use PRETEXT=*text* in a CALL DEFINE statement.

See SAS Note 24324 for examples of these methods.

Common Compute-Block Issues

You should keep the following issues in mind when you are working with a compute block:

- A variable under an across variable can create one too many columns. There is not an automatic way to apply one statement to multiple columns under an across variable. Each column requires its own assignment statement. See SAS Note 23978 for an example of using macro processing to create a compute variable for columns under an across variable.
- A column under an across variable is referenced in the form of *_Cn_*, where n is the column number. The column number is the physical column in the report, including the NOPRINT or NOZERO variables, in the order of the COLUMN statement (left to right).

FREQ Statement

The FREQ statement treats observations as if they appear multiple times in the input data set. The syntax for the FREQ statement is as follows:

FREQ *variable*;

The FREQ statement variable must be a positive integer. If it is not an integer, SAS will truncate it.

FREQ Statement Example:

```
data new;
  set sashelp.class;
  myvar=5;

proc report nowd data=new;
  col name age;
  freq myvar;
run;
```

Output

Name	Age
Alfred	70
Alice	65
Barbara	65
Carol	70
Henry	70

James	60
Jane	60
Janet	75
Jeffrey	65
John	60
Joyce	55
Judy	70
Louise	60
Mary	75
Philip	80
Robert	60
Ronald	75
Thomas	55
William	75

Note: Frequency information is not saved, even when you store a report definition.

RBREAK Statement

The RBREAK statement produces a default summary row at the beginning or end of the detail data. This statement uses the same options as the BREAK statement.

```
proc report nowd data=sashelp.class(obs=5);
  col age, sex, weight;
  define age / group;
  define sex / group;
  break after sex / summarize;
  rbreak after / summarize;
run;
```

WEIGHT Statement

The WEIGHT statement specifies weights for all analysis variables in statistical calculations. The WEIGHT statement syntax is as follows:

WEIGHT *variable*;

The variable does not have to be an integer. If you want to weight a particular variable, you can use the WEIGHT= option, which is used in the DEFINE statement.

The following example illustrates the use of the WEIGHT statement:

```
data new;
  set sashelp.class;
  myvar=5.5;

proc report nowd data=new;
  col name age;
  weight myvar;
run;
```

Output

Name	Age
Alfred	77
Alice	71.5
Barbara	71.5
Carol	77
Henry	77
James	66
Jane	66
Janet	82.5
Jeffrey	71.5
John	66
Joyce	60.5
Judy	77
Louise	66
Mary	82.5
Philip	88
Robert	66
Ronald	82.5
Thomas	60.5
William	82.5

References

Burlew, Michele M. 2005. *SAS Guide to Report Writing: Examples, Second Edition*. Cary, NC: SAS Publishing.

Marcum, Maggie. 2005. SAS Note 23671, "How can I simulate table options from PROC REPORT that are not honored in the ODS HTML destination?" Cary, NC: SAS Institute Inc. Available at support.sas.com/kb/23/671.html.

McMahill, Allison. 2007. *Beyond the Basics: Advanced PROC REPORT Tips and Tricks*. Cary, NC: SAS Institute Inc. Available at support.sas.com/rnd/papers/sgf07/sgf2007-report.pdf.

McMahill, Allison. 2005. SAS Note 24324: "How can I add text at the RBREAK level for a numeric GROUP or ORDER variable?" Cary, NC: SAS Institute Inc. Available at support.sas.com/kb/24/324.html.

McMahill, Allison. 2004. SAS Note 23978, "How can I automate the creation of a COMPUTE variable under an ACROSS variable?" Cary, NC: SAS Institute Inc. Available at support.sas.com/kb/23/978.html.

SAS Institute Inc. 2002-2007. "The REPORT Procedure" in *SAS 9.1.3 OnlineDoc*. SAS Institute Inc. Cary: NC. Available at support.sas.com/onlinedoc/913/docMainpage.jsp.

Tilanus, Erik W. 1997. *Working with the SAS System*. Cary, NC: SAS Publishing.

