In The Compute Block:
Issues Associated with Using and Naming Variables

Arthur L. Carpenter
California Occidental Consultants

ABSTRACT
In the compute block of the PROC REPORT step variables can be both created and used. They may be on the incoming data table, they may be computed in the REPORT step, they can be statistical summaries, and they may even be artificially created by the REPORT process. In the DATA step each variable has a name and that variable's name is used whenever we want to address a particular value on the Program Data Vector. In the REPORT step things are not so simple. The origins and use of a variable will determine how it is to be named in a compute block.

If you do not know how to properly address a variable or report item in the compute block, the compute block will often fail. This paper will discuss the various variable types, how they are used, and how they are addressed in the compute block. Once introduced the rules are fairly easy, and a full understanding of the naming conventions and how they are applied - "use what name when", will go a long way to allowing you to create more complex and successful compute blocks in PROC REPORT.

KEYWORDS
PROC REPORT, Compute Block, Computed variables, Composite names, Report items

INTRODUCTION
The compute block is an extremely valuable tool within PROC REPORT, and since it supports the use of many of the programming statements and functions of the DATA step, it can be used to create columns and manipulate data. To fully use this power we need to be able to access the incoming variables, as well as, columns and statistics created in the step itself. These columns and variables have names and as the naming conventions are **not** as straight forward as in the DATA step, a more complete understanding of how to access them is required.

The following simple compute block writes the value of the variable REGION using the format $REGNAME. This compute block is referencing the report item, REGION, which is a variable on the incoming data table.

```
compute before region;
   line @3 region $regname8.;
endcomp;
```

There are four ways to reference a variable in a compute block, and you can reference any report item that forms a column, even columns that are not printed. In the LINE statement of the above compute block, the variable REGION, which is also a report item and a GROUP variable, is referenced directly (explicitly) by name.

You can reference report items:

- **explicitly by name**
  The variable name can be used directly, as in the above example, when the variable has a define type of GROUP, ORDER, COMPUTED, or DISPLAY. DATA step variables, variables that are created and only used in a compute block, are also always addressed by variable name.

- **using a compound name**
  Compound variable names are needed, when a variable with an ANALYSIS define type has been used to calculate a statistic. The compound name is combination of the variable name and the statistic that it has

been used to calculate. The general form is *variablename.statistic*, and an example of a compound name might be:

```
wt.mean
```

- **by specifying an alias**
  Aliases are specified directly in a compute block. An alias can be specified in the COLUMN statement when you want to use a single analysis variable in more than one way - generally to calculate more than one statistic. The following COLUMN statement generates a series of aliases for the HT analysis variable.

```
columns region ht
      ht=htmin ht=htmax
      ht=htmean ht=htmedian;
```

  In the compute block the alias is addressed directly as in:

```
compute after;
      line @3 'Minimum height is ' htmin 6.1;
endcomp;
```

- **directly using the report column number**
  Sometimes as the report is constructed a given column may not have a specific name. This is especially the case when a variable, with the define type of ACROSS, creates a series of columns. These, and indeed any column in the report, can be referenced by using the column number as an indirect column reference. This pseudo variable name is always of the form

```
_Cxx_
```

  where the *xx* is the column number as read from left to right on the report. The column count **should** include any columns that will ultimately not print *e.g.* those columns defined with NOPRINT or NOZERO.

The REPORT step also creates an automatic temporary variable named _BREAK_. This variable is addressed directly and is available during the execution of the compute block.

DIRECT VARIABLE NAME REFERENCE

When a variable has the define type of DISPLAY, that variable's name is used explicitly in compute blocks. This is shown in the following example which calculates a computed column.

The Body Mass Index, BMI, is a rough measure of health, and for most adults a BMI between 18.5 and 24.9 is generally considered to be in the normal range. The following example calculates the BMI for the students in the SASHELP.CLASS data set.

```
title1 'Extending Compute Blocks';
title2 'Using Variable Names';

proc report data=sashelp.class nofs;
   column name weight height BMI;
   define name / display;
   define weight / display;
   define height / display;
   define bmi / computed format=4.1 'BMI';
   compute bmi;
      bmi = weight / (height*height) * 703;
   endcomp;
```

```
run;
```

```
Extending Compute Blocks
Using Variable Names

  Name          Weight     Height    BMI
  Alfred         112.5         69   16.6
  Alice             84       56.5   18.5
  Barbara           98       65.3   16.2
  Carol          102.5       62.8   18.3
  Henry          102.5       63.5   17.9
  James             83       57.3   17.8
  Jane            84.5       59.8   16.6
  Janet          112.5       62.5   20.2
  Jeffrey           84       62.5   15.1
  John            99.5         59   20.1
  Joyce           50.5       51.3   13.5
  Judy              90       64.3   15.3
  Louise            77       56.3   17.1
  Mary             112       66.5   17.8
  Philip           150         72   20.3
  Robert           128       64.8   21.4
  Ronald           133         67   20.8
  Thomas            85       57.5   18.1
  William          112       66.5   17.8
```

In this example the two variables HEIGHT and WEIGHT have a define type of DISPLAY. When DISPLAY variables are used in a compute block they are referenced directly.

If these same variables had been designated as ANALYSIS (which is typical, and the default for numeric variables without DEFINE statements), the previous compute block would not have worked (see the same example in the next section using these variables with a define type of ANALYSIS). The LOG would contain the following message, which seems to be wrong and is therefore confusing.

```
NOTE: Variable height is uninitialized.
NOTE: Variable weight is uninitialized.
NOTE: Division by zero detected at line 1 column 15.
```

Of course the variables HEIGHT and WEIGHT do exist and are initialized, but they have different names in the compute block when they have a type of ANALYSIS.

The following example converts the weight of each class member from pounds to kilograms. Notice that the compute block is associated with a variable with a type of DISPLAY.

```
title1 'Extending Compute Blocks';
title2 'Using Variable Names';
title3 'Converting Pounds to Kilograms';

proc report data=sashelp.class nofs;
  column name sex ('Weight' weight);
  define name   / order 'Name';
  define sex    / display 'Sex';
  define weight / display format=6. 'Kg';
  compute weight;
    weight = weight / 2.2;
  endcomp;
  run;
```

The compute block will execute for each row of the report, and since WEIGHT has a define type of DISPLAY a direct variable reference is used in the compute block to name the variable to be converted.

```
Extending Compute Blocks
Using Variable Names
Converting Pounds to Kilograms

            S
            e  Weight
  Name      x      Kg
  Alfred    M      51
  Alice     F      38
  Barbara   F      45
  Carol     F      47
  Henry     M      47
  James     M      38
  Jane      F      38
  Janet     F      51
. . . Portions of the report are not shown . . .
```

DATA step variables are always created in compute blocks and are always addressed explicitly using the variable name. This is demonstrated in the following example that counts observations using DATA step variables.

One of the default features of PROC PRINT is the OBS column. The following example adds this observation counter for PROC REPORT output. In the compute block where the OBS column is computed two variables are used. The variable CNT does not appear on the COLUMN statement and is therefore a DATA step variable. Its value will be retained from row to row, consequently a SUM statement is especially useful ❶. The value of CNT, which is the current row number, is assigned to the computed variable OBS ❷.

```
title1 'Extending Compute Blocks';
title2 'Using Variable Names';
title3 'Creating an OBS Column';

proc report data=sashelp.class nofs;
  column obs name sex weight;
  define obs    / computed 'Obs' format=3.;
  define name   / order 'Name';
  define sex    / display 'Sex';
  define weight / display format=6.2 'Weight';
  compute obs;
     cnt + 1;
     obs = cnt;
  endcomp;
  run;
```

The resulting table becomes:

```
Extending Compute Blocks
Using Variable Names
Creating an OBS Column

                 S
                 e
  Obs  Name      x  Weight
    1  Alfred    M  112.50
    2  Alice     F   84.00
    3  Barbara   F   98.00
    4  Carol     F  102.50
. . . Portions of the report are not shown . . .
```

Notice that the computed variable OBS and the Data Step variable CNT are both addressed specifically.

FYI - The problem of the generation of the OBS column is specifically addressed by Pass (2000).

COMPOUND VARIABLE NAMES

In the first example of the previous section, the BMI is calculated for the students in SASHELP.CLASS. In that example the variables WEIGHT and HEIGHT received a define type of DISPLAY and the variables were address explicitly using their names. When the define type of ANALYSIS is used, the variables cannot be address explicitly, instead the use of compound names is required.

```
title1 'Extending Compute Blocks';
title2 'Using Compound Variable Names';

proc report data=sashelp.class nofs;
   column name weight height BMI;
   define name / display;
   define weight / analysis;
   define height / analysis;
   define bmi / computed format=4.1 'BMI';
   compute bmi;
      bmi = weight.sum / (height.sum*height.sum) * 703;
   endcomp;
   run;
```

A compound name is always going to be a combination of the variable name and a statistic. When a statistic has not been specified, the statistic will be the SUM.

```
Extending Compute Blocks
Using Compound Variable Names

  Name          Weight     Height    BMI
  Alfred         112.5         69   16.6
  Alice             84       56.5   18.5
  Barbara           98       65.3   16.2
  Carol          102.5       62.8   18.3
  Henry          102.5       63.5   17.9
    . . . Portions of the report are not shown . . .
```

It is because HEIGHT and WEIGHT are **analysis** variables, that compound names are required in the compute block. Whereas the following assignment statement worked in the first example of the previous section, it would

not have worked here, because compound names are not being used.

```
compute bmi;
   bmi = weight / (height*height) * 703;
endcomp;
```

USING AN ALIAS AS A COLUMN REFERENCE

A variable alias is created on the COLUMN statement, and once created, each alias can be associated with a DEFINE statement. When aliases are used in a compute block they are addressed using the alias name and not the original variable's name.

In the following example the alias WTMAX, which is created with the specification `WT=WTMAX` in the COLUMN statement, allows the you to use the report item WT in two different DEFINE statements. This alias can then be used directly in the compute block.

```
title1 'Using an Alias';

proc report data=sashelp.class nofs;
   column sex ('Weight in Pounds' weight weight=wtmax wt_range);
   define sex      / group format=$3.;
   define weight   / analysis min 'Min' format=5.1;
   define wtmax    / analysis max 'Max' format=5.1;
   define wt_range / computed 'Range' format=5.1;
   compute wt_range;
      wt_range = wtmax - weight.min; ❶
   endcomp;
   run;
```

```
     Using an Alias

       Weight in Pounds
Sex     Min     Max  Range
F      50.5   112.5   62.0
M      83.0   150.0   67.0
```

Notice the use of the compound name `weight.min` in the compute block assignment statement ❶. Because WTMAX is an alias, the following assignment statement would not work:

```
wt_range = wtmax.max - wt.min;
```

Nor could we ignore the compound name for the analysis variable WT. The following assignment statement would also fail.

```
wt_range = wtmax - wt;
```

USING INDIRECT COLUMN REFERENCES

Derived columns, such as those created using the ACROSS define type, will not have a column name on the COLUMN statement. When these columns are used in a compute block, an indirect column reference is required.

In the following example the ratio of the mean weight of the two genders is calculated. Since we want the weight of the females to be the numerator we use **`_c3_ / _c2_`** as the ratio.

```
title1 'Using Indirect Column References';
```

```
proc format;
  value $gender
    'M' = 'Male'
    'F' = 'Female';
  run;

proc report data=sashelp.class nofs split='*';
  column age ('Mean Weight*in Pounds' sex,weight ratio);
  define age    / group width=10 'Age';
  define sex    / across 'Gender'
                  format=$gender.
                  order=data;
  define weight / analysis mean format=6. ' ';
  define ratio  / computed format=6.3 'Ratio*F/M ';
  rbreak after  / skip summarize;
  compute ratio;
     ratio = _c3_ / _c2_;
  endcomp;
  run;
```

We verify that the weight for males is in the second column (_C2_) and that the weight of the females is in the third column (_C3_).

Using Indirect Column References

Age	Mean Weight in Pounds Gender Male	Female	Ratio F/M
11	85	51	0.594
12	104	81	0.780
13	84	91	1.083
14	108	96	0.895
15	123	112	0.916
16	150	.	.
	109	90	0.827

Since the column numbers of the table are used, the programmer must have a certain knowledge of the final table before selecting the numbers used in the indirect reference. In the previous example we know that in the incoming data table the first observation is for a male patient and that the ORDER= option has been set to DATA. Changing the DEFINE statement for SEX to use ORDER=FORMATTED

```
define sex / across 'Gender'
           format=$gender. order=formatted;
```

would place FEMALES in column 2 and the assignment statement for RATIO would be:

```
compute ratio;
   ratio = _c2_ / _c3_;
endcomp;
```

USING THE AUTOMATIC TEMPORARY VARIABLE _BREAK_

As PROC REPORT executes, the data is summarized and held in memory before being transferred to the report itself. One of the columns in this intermediate summary is the automatic variable _BREAK_. This column is used to identify lines were generated as a result of grouping variables, BREAK, or RBREAK statements.

Adding an OUT= option to the PROC statement in the previous example allows us to get a glimpse of this temporary table.

```
proc report data=sashelp.class nofs split='*'
            out=temptbl;
 . . . . statements not shown . . . . .
   run;

proc print data=temptbl;
   run;
```

The resulting data table is shown below. _BREAK_ is a character variable and can be addressed directly in the compute block. It has missing values for the detail rows of the report, however for the line that is generated for the report break, the variable _BREAK_ is equal to '_RBREAK_'. Notice that AGE is missing for the summary line.

Using Indirect Column References

Obs	Age	_C2_	_C3_	ratio	_BREAK_
1	11	85.00	50.500	0.59412	
2	12	103.50	80.750	0.78019	
3	13	84.00	91.000	1.08333	
4	14	107.50	96.250	0.89535	
5	15	122.50	112.250	0.91633	
6	16	150.00	.	.	
7	.	108.95	90.111	0.82709	_RBREAK_

The _BREAK_ variable can be very valuable when using logic statements within compute blocks that need to detect group boundaries.

SUMMARY

Depending on the type of variable and how it is used there are different ways to name variables within the PROC REPORT's compute block:

- Direct reference for DISPLAY variables, variable aliases, and DATA step variables
- Compound names for ANALYSIS variables and their associated statistics
- Indirect naming of columns by report column number

There is also the special temporary variable, _BREAK_, which is available for use in programming statements within the compute block. This variable is also addressed directly by name.

ABOUT THE AUTHOR

Art Carpenter's publications list includes three books, and numerous papers and posters presented at SUGI and other user group conferences. Art has been using SAS® since 1976 and has served in various leadership positions in local, regional, national, and international user groups. He is a SAS Certified Professional™ and

through California Occidental Consultants he teaches SAS courses and provides contract SAS programming support nationwide.

AUTHOR CONTACT

Arthur L. Carpenter
California Occidental Consultants
P.O. Box 430
Vista, CA 92085-0430

(760) 945-0613
art@caloxy.com
www.caloxy.com

REFERENCES

Pass, Ray, 2000, "PROC REPORT - Land of the Missing OBS Column", *Proceedings of the Twenty-fifth Annual SAS® Users Group International Conference*, Cary, NC: SAS Institute Inc., paper 105-25, pages 559-560.

Portions of this paper were borrowed, *with permission*, from Arthur L. Carpenter's forthcoming book on PROC REPORT.

TRADEMARK INFORMATION